University of Helsinki
Department of Digital Humanities
Language Technology

Master's thesis

Using POS n-grams to detect grammatical errors in Finnish text

Mikko Aulamo
014151453

Supervisor: Jörg Tiedemann 20.10.2019

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Tiedekunta/Osasto – Fakultet/Sektion – Faculty Humanistinen tiedekunta		
Tekijä – Författare – Author Mikko Aulamo		
Työn nimi – Arbetets titel – Title Using POS n-grams to detect grammatical errors in Finnish text		
Oppiaine – Läroämne – Subject Kieliteknologia		
Työn laji – Arbetets art – Level Pro gradu	Aika – Datum – Month and year 10 / 2019	Sivumäärä– Sidoantal – Number of pages 67

Tiivistelmä – Referat – Abstract

Automaattinen kieliopin tarkistus on hyödyllinen työkalu henkilöille, jotka kirjoittavat julkaistavia tekstejä. Kieliopintarkistimista on myös hyötyä kielenoppijoille. Suomen kielelle tehdyt käytetyimmät tarkistimet ovat sääntöpohjaisia, minkä vuoksi ne kattavat vain pienen osan kielioppivirheistä, ja sääntöjoukon laajentaminen vaati paljon käsintehtävää työtä. Tilastollisilla menetelmillä voidaan löytää suurempi määrä eri virheitä ilman käsinlaadittavia sääntöjä. Eräs helposti toteutettavissa oleva tilastollinen tapa on kerätä esimerkkijoukko kieliopillisia n-grammeja, ja verrata, löytyykö tarkistettavan lauseen kaikki n-grammit esimerkkijoukosta. Suomen kielessä on paljon taivutusmuotoja, ja uusia sanoja pystytään myös luomaan käyttämällä johtimia. Jos n-grammien yksikköinä käytetään saneita, esimerkkijoukon tulee olla käsittämättömän suuri, jotta se voi kuvata Suomen kieliopin kattavasti. Tämä pro gradu -työ esittää kieliopintarkistusmetodin, joka on helppo toteuttaa, koska siinä käytetään n-grammeja yllä mainitulla tavalla, mutta n-grammien yksikköinä käytetään part-of-speech (POS) -informaatiota saneiden sijaan, jolloin esimerkkijoukon n-grammit on mahdollista kerätä, ja niiden määrä pysyy tarpeeksi pienenä käsiteltäväksi. N-grammit ja niiden esiintymäkertojen lukumäärät kerätään suomenkielisestä morfologisesti annotoidusta FinnTreeBank -korpuksesta.

Kieliopintarkistin arvioidaan 200 eri koeasetelmassa, jotka eroavat toisistaan viidellä eri tavalla. Puolet tarkistimista koulutetaan pienellä käsinannotoidulla korpuksella ja puolet suurella automaattisesti annotoidulla korpuksella. Puolet tarkistimista käyttää lauserajamerkintöjä n-grammeissaan ja puolet ei. Puolissa asetelmissa valitaan yksi lauserakenteen tulkinta tarkistettavaksi, ja puolissa tarkistetaan kaikki mahdolliset rakennetulkinnat. Jokainen tarkistimista käyttää myös yhtä viidestä esiintymäkertojen raja-arvoista, joka n-grammien tulee ylittää, jotta ne hyväksytään kieliopillisiksi. Lisäksi jokainen tarkistimista käyttää yhtä viidestä POS n-grammityypistä, joista jokainen sisältää eri yhdistelmän POS-informaatiota. Kieliopintarkistin arvioidaan konekäännösjärjestelmän tuottamilla kieliopillisesti virheellisillä lauseilla sekä niiden kieliopillisesti oikeilla vastineilla. Suurimmassa osassa koeasetelmia tarkistin merkitsee vain vähän virheitä ja on usein väärässä, tai tarkistin merkitsee lähes kaikki lauseet, myös kieliopilliset, virheellisiksi. Tarkkuuden kannalta parhaiten suoriutuneessa asetelmassa käytetään suurta korpusta, ei lauserajamerkintöjä, kaikki lauserakennetulkinnat tarkistavaa metodia, pientä esiintymäkertaraja-arvoa ja POS-informaatiota, jolla on vähiten mahdollisia esiintymämuotoja. Tässä asetelmassa tarkistin on noin 86% kerroista oikeassa merkitessään kielioppivirheitä, mutta toisaalta se löytää vain noin 27% testiaineiston virheistä. Toteutettu metodi ei siis sellaisenaan ole toimivia Suomen kieliopin tarkastamiseen, mutta metodia voisi parantaa lisäämällä siihen disambiguaatiokomponentin ja käyttämällä suurempaa koulutuskorpusta.

Avainsanat – Nyckelord – Keywords
grammar error detection, grammar checking, part of speech, n-gram

Säilytyspaikka – Förvaringställe – Where deposited
Keskustakampuksen kirjasto

Muita tietoja – Övriga uppgifter – Additional information

Contents

1 Introduction

Automatic grammar checking is a useful tool for anyone who writes texts that need to be published. These texts can be anything from job applications and business texts to organization-wide announcements and journalistic texts and beyond. Time is usually a constraint when working with large amounts of text, and automatic grammar checkers can drastically reduce the time spent on ensuring proper language. The development of grammar checking tools has mainly focused on checking the English language, although there is also clearly a need for these tools for smaller languages, like Finnish. The purpose of this thesis is to evaluate whether part of speech based n-grams could be used to detect grammar errors in Finnish text.

1.1 Need for grammar checkers

Using proper language is important when writing academic texts. Correct grammar ensures that one's ideas and thoughts come out clearly. In their work, Cavaleri and Dianati (2016) conduct a grammar survey for 18 English speaking students from two different Australian colleges. The survey charts the students' confidence in their grammar skills and their experience with Grammarly, an online writing assistant[1]. On a five grade scale ("Strongly disagree", "Disagree", "Neutral", "Agree", "Strongly agree"), 9 of the 18 students "agreed" with the statement "I do not always feel confident that I have written correct sentences" and 7 "agree" with the statement "I am fine with English grammar, but I find it difficult to express my ideas in writing". Both of these are cases where Grammarly can provide assistance. When evaluating the usefulness of Grammarly on a scale from 0 ("Not useful at all") to 5 ("Extremely useful"), 9 students give the rating 5 and 6 students give the rating 4. There seems to be a demand for grammar checking tools among young academics at least for English text. Although most scientific publications are written in English, it is also meaningful to be able to present one's research and ideas to colleagues and students in one's native language, for example Finnish, which is the language that the work in this thesis focuses on.

Grammar checking is also a useful language learning tool. Non-native language speakers often produce text that contains grammar errors, which is a natural part of learning a new language. It is crucial for the learner that these errors are detected and corrected. Scanning through a text and finding errors by hand is time-consuming and laborious. This is where automatic

[1] `https://www.grammarly.com`

grammar checking can prove useful. According to Soni and Thakur (2018), there are over 600 million speakers of English as a second language or English as a foreign language, who would benefit from regular grammar checking of their texts and automating this process would be the most efficient method. There is a similar need for grammar checking for Finnish learners, although on a smaller scale. According to Miettinen and Helamaa (2019), the amount of immigration in Finland has increased significantly since the year 2000, with the peak numbers being over 30000 immigrants annually during the years 2012-2014 and 2016. Heikkilä and Peltonen (2002) concluded in their work, that Finnish language proficiency is one of the key factors in immigrants integrating into Finnish society. Reaching employment often depends on adequate language skills. They also mention that especially immigrants from Russia wish to have more language training and further language teaching in their field of profession. This indicates that there is a rising need for more Finnish teaching, and similarly, the demand for language learning tools, including automatic grammar checking, is on the rise.

Grammar checkers have been used in aiding machine translation. Mitamura and Nyberg (1995) use a grammar checking tool as a first step of machine translation in their KANT system. They translate from English to French, and the role of grammar checking is to make sure that the English source sentences are proper language. Sågvall Hein (1997) uses a similar approach in her work on Multra, a machine translation system, that can translate from Swedish to English and German. The development of Multra included the development of a grammar checker, which checks all the Swedish source sentences and generates their grammatical structures, which can then be forwarded to other components of the machine translation system. Stymne and Ahrenberg (2010) use grammar checking for postprocessing of statistical machine translation (SMT) systems' output and for evaluating the sentences translated with SMT. SMT produces often ungrammatical translations and the authors tackle this issue by applying corrections suggested by a grammar checker to the translated sentences. Stymne and Ahrenberg, also utilize grammar checking in the evaluation phase and find that evaluating the translations with a grammar checker gives useful information on grammatical correctness when used in addition to the standard Bleu metric (Papineni et al., 2002).

1.2 Main approaches to grammar checking

A text is considered to have correct grammar if the language is syntactically correct. Grammar errors are introduced when the syntax is broken, for example with incorrect inflection. Spelling errors are distinct from grammar errors, but a text containing spelling errors is also ungrammatical, as the errors create unknown words that break the syntax.

Grammar checking is a challenging task. Language is inherently ambiguous and many of these ambiguities can be resolved only based on the surrounding context. Domeij et al. (2000) point out that errors caused by ambiguity are difficult to detect for their Swedish grammar error detector Granska, which is based on error detection rules and probabilistic methods. Hagen et al. (2001) list other challenging problems that they came across when building a rule-based grammar checker for Norwegian. These problems include non-standard inflection, misspelled words that are homographous with valid words and handling missing words. Similarly, Deksne and Skadiņš (2011) present issues which existed in previous grammar checkers for Latvian and which led them to work on a new checker. According to them, some of the most pressing issues are recognizing errors over long distances, complex syntax errors and morphological ambiguity.

Historically, there are two main approaches for grammar checking: rule-based methods and statistical methods. The idea in rule-based methods is to find a syntactical representation for an input sentence and then to apply error detection rules to the sentence. These error rules are handwritten, which makes it very laborious to implement a robust system that could cover most grammar errors in a language. The advantage of rule-based systems is that they are very good at detecting the type of errors that the rules do cover. In statistical methods, the grammar checker is trained on language data. In a sense, the system formulates the error rules based on the language that it sees in the training data. This makes implementing a statistical system easier than a rule-based one since it requires much less work by hand. On the other hand, the training data needs to be carefully chosen to represent a given domain, otherwise, the system could leave some error cases undetected because the error rules are not explicitly specified. Rule-based and statistical methods are described in more detail in section 2.

There are some existing grammar checkers for Finnish but they seem to detect errors in a very limited spectrum. Voikko[2] is an open-source language tool package that includes a grammar checking feature. Voikko is a rule-based

[2] `https://voikko.puimula.org/gchelp/fi/`

checker that detects 18 commonly made mistakes. Over half of the rules are tailored to find stylistic errors, e.g. proper punctuation and capitalization, which can be considered to be part of grammar but they do not really evaluate the correctness of linguistic structures. The rest detect grammatical errors in the language itself. For example, the checker detects whether there is agreement between a negative verb and the main verb and whether the main verb is missing. Microsoft Office program package[3] also has a grammar checking feature for Finnish. It is unclear, what kind of methods Microsoft's checker is based on, but in use, it functions very much like the Voikko checker. Punctuation and capitalization as well as some grammatical errors, like doubled words and extra verbs, are detected by both Microsoft and Voikko. However, there are some errors that Voikko does recognize but Microsoft does not, e.g. the aforementioned agreement between a negative verb and the main verb, and a missing main verb. This indicates that Microsoft's checker is probably also a rule-based one but equipped with a different ruleset. Both of these checkers do well when detecting errors that fall under the hand-crafted rules but they completely ignore all other cases. Both seem to have a very limited number of rules and writing new rules to cover most errors in Finnish would be very laborious. To aid in this generalization task, statistical methods could prove useful.

1.3 My proposed work

My work proposes a statistical approach to detecting grammar errors in Finnish text. This method aims to be capable of capturing more general grammar errors than strictly rule-based methods would, while also being easier to implement, since one does not have to write rules for each error case by hand. In this approach, the first step is to extract n-grams and their frequencies from Finnish text. This collection of grammatically correct n-grams will serve as the reference n-gram set. The n-grams are gathered from FinnTreeBank (FTB) (Voutilainen et al., 2012), which is a corpus of Finnish sentences that are annotated with morphological information. Finnish is a morphologically complex language as described by Karlsson (2008), and using n-grams based on actual words would require an impossibly large and varied training corpus for it to be applicable in a grammar detection task. Instead, pieces of morphological (or part of speech) information provided by FTB are used as the units of the n-grams. This allows for larger grammar error coverage but also increases the possibility of mistakes in detecting errors. The

[3] `https://support.office.com/fi-fi/article/oikeinkirjoituksen-ja-kieliopin-tarkistaminen-officessa-5cdeced7-d81d-47de-9096-efd0ee909227`

actual grammar checking of a sentence consists of three main steps: first, each word in the input sentence is morphologically analyzed using Omorfi (Pirinen, 2015), secondly, an interpretation for the whole sentence is chosen and its part of speech (POS) n-grams are gathered, and finally, the system checks whether all these n-grams are found in the reference n-gram set with a sufficient frequency value or not. If all of the POS n-grams are represented in the reference n-gram set, the input sentence passes the correctness test. Otherwise, the sentence is flagged as an error.

The grammar checker is implemented in multiple different setups in order to find the optimal settings and to find out which setups might favor precision over recall or vice versa. Some versions of the checker might also detect different types of errors than others. Two reference n-gram sets are collected from two different versions of FinnTreeBank: FTB1 and FTB3. FTB1 consists of 19000 sentences and is fully annotated by hand, and FTB3 contains over 4 million sentences but it is annotated automatically. FTB1 is likely to have fewer annotation errors and is smaller than FTB3. Thus it is likely to be stricter than FTB3, resulting in more errors being flagged and higher recall value. Conversely, FTB3 will pass some ungrammatical sentences as correct, and when it does flag an error, it will be certain of its decision, leading to higher precision. Each of the FTB n-gram sets has two versions: one where the sentence borders are marked with tags (WB) and one where they are not (WoB). Essentially, having tags for borders increases the number of unique n-grams. This should lead to the checker detecting more types of errors with regard to how sentences can start or end. Setups using WB n-gram set are expected to flag more errors and to have higher recall than WoB setups, which, in turn, are likely to have higher precision, as they flag errors more rarely. Two different error detection algorithms are implemented: one that checks the most probable (CMP) morphological interpretations of a sentence and one that checks all possible (CAP) interpretations of a sentence. CMP is a naive approach, where the POS tag for each word of an input sentence is chosen based on what is the POS tag that occurs most frequently for that word in the FTB1 corpus. CAP is an exhaustive method, that checks all morphological interpretations for a sentence. The interpretations are formulated by creating all possible combinations of POS tags that are provided by Omorfi. Each of the interpretations is individually checked and if an interpretation without error is found, the sentence is considered grammatically correct. There are more chances for a given sentence to be passed as grammatical using CAP, and thus it should have higher precision but lower recall than CMP. The grammar errors are also detected with different n-gram frequency cutoff points. If a POS n-gram's frequency value in an n-gram set

does not reach a certain threshold, it is not considered grammatical. Lower cutoff points allow more sentences to pass the error detection test than higher cutoff points, which should lead to lower cutoffs having higher precision but lower recall. Each reference n-gram set has different versions of POS-tags carrying different amounts and different types of morphological information. The expectation is that the less information the tags have, the more sentences will pass the error detection test, which leads to higher precision and lower recall. There are twelve error types that are looked at with a special interest focusing on two of them: a word in a sentence has the wrong grammatical number and a word in a sentence has the wrong grammatical case. The hypothesis is that when the POS-tags contain information on number, the system should be able to detect number errors but not case errors. Similarly, tags containing information on case should be able to detect case errors but not number errors.

In section 2, the history and theory of grammar checking methods are described in further detail. The data used in this work, as well as the n-gram extraction process, are described in section 3. Section 4 explains the grammar error detection methods used in the grammar checker implementation of this work. The test data and the evaluation setups for assessing the grammar checker are described in section 5. The results of the evaluation are displayed and their implications are analyzed in section 6. Finally, the work as a whole is discussed and concluded in section 7.

2 Theory

There are two main approaches to grammar checking that were considered when implementing the error detection system in this thesis: rule-based methods and statistical methods. There are advantages and shortcomings to both approaches. Rule-based systems can be tailored by hand to detect very specific errors, but implementing a large scale grammar checker requires a huge amount of labor. Statistical checkers are trained on existing language material and are less labor-intensive. However, one needs to make sure that the training corpus is suitable for the grammar checking system, and in some cases, the language data has to be morphologically or syntactically analyzed and tagged, which is already a task by itself.

2.1 History

Using rule-based methods for grammar checking started in the 1980s. In a basic approach to rule-based checking, the sentence to be checked is first transformed to a parsed representation and error rules are then applied to this presentation to see whether the sentence is grammatical. Macdonald (1983) describes some early tools designed to help in writing texts. These tools include grammar checking programs that are based on handwritten error rules, e.g. checking for correct punctuation and finding doubled words. Other programs shown in the paper help with stylistic features and spell checking. Heidorn et al. (1982) introduce EPISTLE, a text critiquing system, which includes a rule-based grammar checking feature. EPISTLE grammar checker parses an input text using approximately 250 rules. The rules are tailored to find the most frequently mentioned errors in English usage literature and the most commonly made errors in business correspondences. Such errors include subject-verb disagreement, using the wrong pronoun case and noun-modifier disagreement. Another early rule-based grammar checker system is CRITIQUE (Richardson and Braden-Harder, 1988), which is an extension of EPISTLE. Like EPISTLE, CRITIQUE parses text according to grammar rules, but the software is better optimized and it runs on multiple processors. CRITIQUE's grammar rules are gathered from an office correspondence database and from over 3000 pages of text submitted by users in the feedback phase. A more recent example of a rule-based checker is by Singh et al. (2016). They admit that a major drawback in using rule-based methods is that there will inevitably be errors which the checker cannot detect. A checker that covers all possible errors in a language would require handwritten rules for all error cases, which is unfeasible.

Statistical methods for grammar checking are easier to implement than rule-based methods. One does not have to write each error rule by hand. Instead, the checker is trained on language data and the checker decides whether a structure is grammatically wrong based on what it has seen in the training data. This also means that the checker can be suited for different language domains depending on what kind of material the checker is trained on. One of the earliest works in statistical grammar checking is by Atwell (1987), whose CLAWS grammar checker for English is based on a constituent likelihood grammar, which gives information on how likely different language structures are to appear after one another. CLAWS uses a simple Markovian model for word-tagging, which is much more efficient than parsing whole multi-level constituent-structure trees. Nazar and Renau (2012) present their grammar checker for Spanish, in which they utilize the Google Books N-gram

Corpus, which consists of n-grams up to length five and their frequencies in books published mostly after the year 1970. The basic principle of their work is that if the n-grams of an input sentence occur frequently enough in the corpus, the input sentence is grammatically correct. Using n-grams consisting of plain words can cause issues in this method since the reference corpus cannot contain all possible n-grams with all possible word combinations. Nazar and Renau use word n-grams and handle the missing word cases by ignoring the n-gram sequences which contain words not found in the corpus. Additionally, the Google Books N-gram Corpus contains very extensive amounts of Spanish data, which helps to keep the number of unknown words down. Handling missing words in this manner is not possible if the grammar checker determines correctness by calculating word sequence probabilities using multiplication of word frequencies. A sequence containing an unknown word, i.e. a word with frequency 0, would always result in the whole sequence to have the probability 0. This is the case in the checker for Bangla by Alam et al. (2007). They resolve this issue by using n-grams consisting of POS tags instead of words. This approach requires the addition of a POS tagger to the checker pipeline.

2.2 Language models

Language models are the basis for grammar checking as well as many other applications. A language model is typically a model, that assigns probabilities to word sequences. There are two main types of language models: count-based and continuous-space. Count-based models estimate probabilities by counting n-grams and use smoothing techniques to overcome issues with unseen n-grams (Saul and Pereira, 1997). Count-based language models have been used in a variety of different tasks, including speech recognition (Katz, 1987; Kuhn and Mori, 1990), statistical machine translation (Koehn et al., 2003; Chiang, 2005), POS-tagging (Weischedel et al., 1993; Merialdo, 1994), information retrieval (Ponte and Croft, 1998; Miller et al., 1999) and grammar checking (Chodorow and Leacock, 2000; Lin et al., 2011). Continuous-space language models, or neural language models, solve the issue of data sparsity by representing words as vectors. Semantically similar words are represented by n-gram model vectors that are close to each other in their vector space (Bengio et al., 2003). Neural models can be applied to the same tasks as their count-based counterparts: speech recognition (Graves et al., 2013; Graves and Jaitly, 2014), neural machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014), POS-tagging (Schmid, 1994; Andor et al., 2016), information retrieval (Shu and Kak, 1999; Liu et al., 2015) and

grammar checking (Xie et al., 2016; Liu and Liu, 2017).

Count-based n-gram language models can be constructed to answer questions with the following form: what is the probability of a word given a string of preceding words. For example, what is the probability of the word "sunny" appearing after "the weather is going to be", which can be represented as P(*sunny*|*the weather is going to be*). Martin and Jurafsky (2009) explain, that a simple way to calculate this probability is to use maximum likelihood estimations based on counting the number of the occurrences of the phrase "the weather is going to be sunny" and divide it by the number of the occurrences of the phrase "the weather is going to be" in a representative corpus. Martin and Jurafsky continue with more sophisticated and efficient methods for calculating the probabilities, for example using the chain probability rule, where the probability P(*it is raining*) is split up as follows: P(*it*)P(*is*|*it*)P(*raining*|*it is*). Realistically, the training corpus for the language model cannot always contain all possible valid word sequences, which causes the model to output zero probabilities for sentences that contain words that were not encountered in training. A simple approach to tackle this issue is to convert all unknown words to the unknown word token <UNK> and then calculate the probabilities for <UNK> as if it was a regular word, as described by Martin and Jurafsky.

However, we do not need to go this far as the work presented in this thesis utilizes a simplified language model, or a reference n-gram set, similar to the one used by Nazar and Renau (2012). Their language model is the whole Google Books N-gram corpus, and in order to determine whether sentences contain errors or not, they only need to know if a given n-gram in an input sentence appears in the Google Books corpus frequently enough. Bigert and Knutsson (2002) detect context-sensitive spelling errors using reference POS trigrams derived from an annotated corpus, and comparing the trigrams of an input sentence to the reference trigrams. In order to avoid false positives, rare POS tags are first replaced with more common representative tags, and rare n-grams are transformed into more frequent ones. If the n-gram still has a low frequency after the transformations, the n-gram is considered to be erroneous. Sjöbergh (2005) introduces an approach where only a chunker and unannotated text are needed to conduct grammar error detection. First, an unannotated training corpus is fed through the chunker, resulting in a collection grammatically correct phrase chunks. In the error detection phase, an input sentence is chunked, and if the chunks do not appear frequently enough in the correct collection of phrases, the chunk is considered to be ungrammatical. Sjöbergh (2009) also experiments with using the internet as a reference corpus. Bigrams from an input text are sent to a search engine,

and if a bigram gives no results, it is reported as an error.

2.3 Part of speech

Part of speech (POS) is a class to which a group of words with features similar to each other can be assigned. POS category can be based on word class (noun, adjective, verb...), number (singular, plural...), person (first, second, third...), tense (past, present, future...) and case (genitive, accusative...) among other features. Using POS-based language models is beneficial in a variety of tasks. Integrating POS tags to speech recognition models improve their performance (Heeman, 1999; Collins et al., 2005). POS-based models help with the issue of long-range word reordering in statistical machine translation (Rottmann and Vogel, 2007; Niehues and Kolss, 2009). Pla et al. (2001) use stochastic semantic language models based on semantic units and POS tags to construct the language understanding component of a Spanish railway timetable query dialogue system.

Finnish has 15 case endings, which is more than most European languages have. Finnish morphology allows extensive derivation of word endings to form independent words. Words can have multiple word endings after each other, e.g. derivational endings, case endings, possessive suffixes, and particles (Karlsson, 2008). Thus, Finnish text is bound to contain copious amounts of unique word tokens. Comprising a reference set of n-grams would require an unrealistically large and diverse training corpus. To circumvent this, part of speech information, rather than the actual words, can be used as units of n-grams. This generalization captures a more broad view of the language with less training data. Brown et al. (1992) point out that some words are close to each other in syntactical function and meaning, and by assigning classes to groups of similar words, one can create compact language models with capabilities to predict previously unencountered sequences. Niesler and Woodland (1996) show that using word-category based language models allows generalization of unseen word sequences and reduced sparseness of the data when compared to word-based models. Niesler and Woodland apply their model to a tagging task and the performance is on par with conventional n-gram models. POS n-gram based grammar error detection work has been done by Bigert and Knutsson (2002) and Alam et al. (2007).

3 Data

The grammar error detection system in this thesis is based on a simplified language model that is trained on an annotated corpus. In this case, training means gathering a set of all POS n-grams ranging from length 2 to 5 and the number of their occurrences. The language model consists of n-grams and their frequencies in the training corpus, and the model can also be called as the reference n-gram set. For an n-gram to be considered grammatically correct, it has to exist in this set with a sufficient frequency value.

The first step in implementing the proposed grammar error detection system is to find an annotated corpus and to collect the POS n-grams from it. As suggested by previous work, the training data has to represent a wide coverage of Finnish grammar (Alam et al., 2007; Nazar and Renau, 2012). Wide coverage can be achieved by either using a smaller, carefully selected linguistically varied corpus or a huge set of texts, which most likely will contain a comprehensive set of grammatical features. The accuracy of POS annotations in the training corpus is also an issue to be considered. Using human-made manual annotations ensures the best probability of obtaining accurate annotations, but annotating by hand is realistically feasible only on a smaller corpus. Using POS taggers is more likely to produce incorrect interpretations, although, within certain settings, POS taggers can reach almost perfect results (Manning, 2011; Garrette and Baldridge, 2013). POS taggers can be applied to large texts much faster than having humans annotate them, which helps mitigate the problem of inaccurate annotations.

3.1 FinnTreeBank

In this work, FinnTreeBank is used as the training corpus. FinnTreeBank is a corpus of Finnish sentences and sentence fragments that are annotated with syntactical and morphological information. Voutilainen et al. (2012) describe characteristics of FinnTreeBank in their manual. FinnTreeBank 1 (FTB1) contains 19000 sentences and 160000 tokens annotated carefully by hand. All the text examples in FTB1 are linguistic examples from a descriptive grammar of Finnish, VISK (Hakulinen et al., 2004). VISK contains a large variety of different language structures. Thus, training a grammar checker based on FTB1 allows the system to capture a comprehensive view of correct Finnish grammar. FinnTreeBank 3 (FTB3) contains 4367000 sentences and 76369000 tokens annotated automatically. FTB3 consists of publicly available Finnish corpora. FTB3 is not guaranteed to contain all the grammar

examples of VISK, but by being much larger than FTB1, it still has the potential to represent a robust picture of Finnish grammar. Having access to both FTB1 and FTB3 allows the comparison of two versions of the system: one trained on a small human-annotated corpus and one trained on a huge automatically annotated corpus.

3.2 Tag types

Each token in the FinnTreeBank corpus is annotated morphologically. The analysis contains extensive POS information, including word classes, activeness, mood, tense, person, number, participles, infinitives, cases, possessives, and clitics among other features (Voutilainen et al., 2012). The work in this thesis is mainly focused on detecting grammar errors related to number/person and case, and the whole set of features is not required. In FinnTreeBank, nominals have grammatical number while verbs and pronouns have grammatical person, but in this work, they are both referred to as number (or nu), as they carry a similar meaning.

The only morphological features that are looked at are word class, number, and case because the main interest is in detecting number and case-related errors. All word class, number and case components, that are used in the POS-tags, are shown in table 1. FinnTreeBank also contains word class information for auxiliary verbs and proper nouns. They are converted into verbs and nouns respectively because they are used in similar contexts and this helps in keeping the number unique POS-tags down. From each token in FinnTreeBank, five different tag combinations are extracted: POS-tag, whole-tag, num/case-tag, num-tag, and case-tag.

Word-class-tag (wc) contains information on only the token's word class, e.g. "A" or "N".

Whole-tag (wt) contains information on word class, case and number, e.g. "N Gen Pl" or "V Sg3". If the token has no case or number, that information is left out, e.g. "Adv".

Num/case-tag (nc) contains information on case and number but not word class, e.g. "Pos Gen Pl". The word class information is always replaced by the string "Pos" to avoid having empty strings as tags for tokens that have no case or number information. For example, adverbs result into "Pos".

Num-tag (nu) is similar to num/case-tag but it excludes case information, e.g. "Pos Pl".

word classes	abbreviation	explanation
	A	Adjective
	Abbr	Abbreviation
	Adp	Adposition
	Adv	Adverb
	CC	Coordinating conjunction
	CS	Subordination conjunction
	Interj	Interjection
	N	Noun
	Num	Numeral
	Pron	Pronoun
	Pun	Punctuation
	Unknown	Unknown
	V	Verb
cases		
	Abe	Abessive
	Abl	Ablative
	Acc	Accusative
	Ade	Adessive
	All	Allative
	Com	Comitative
	Ela	Elative
	Ess	Essive
	Gen	Genitive
	Ill	Illative
	Ine	Inessive
	Ins	Instructive
	Nom	Nominative
	Ptv	Partitive
	Tra	Translative
numbers		
	Sg	Singular (nominals)
	Pl	Plural (nominal)
	Sg1	Singular 1st person (verbs)
	Sg2	Singular 2nd person (verbs)
	Sg3	Singular 3rd person (verbs)
	Pl1	Plural 1st person (verbs)
	Pl2	Plural 2nd person (verbs)
	Pl3	Plural 3rd person (verbs)

Table 1: All word classes, cases and numbers that appear in POS-tags

Case-tag (ca) is similar to num/case-tag but it excludes number information, e.g. "Pos Gen".

As an example, the tags for the sentence "Koira on iloinen ." are shown in table 2.

	"Koira"	"on"	"iloinen"	"."
Word-class-tag (wc)	N	V	A	Punct
Whole-tag (wt)	N Nom Sg	V Sg3	A Nom Sg	Punct
Num/case-tag (nc)	Pos Nom Sg	Pos Sg3	Pos Nom Sg	Pos
Num-tag (nu)	Pos Sg	Pos Sg3	Pos Sg	Pos
Case-tag (ca)	Pos Nom	Pos	Pos Nom	Pos

Table 2: Example tags for "Koira on iloinen."

3.3 N-gram extraction

N-gram extraction is conducted for FTB1 and FTB3 corpora, and the resulting n-gram sets are kept separate so they can be compared to one another. For both corpora, additional n-gram sets with sentence border tags are created. This allows us to examine how the inclusion of border tags affects the performance of the grammar checker. In total, there are four kinds of main sets: FTB1, FTB3, FTB1 with border tags and FTB3 with border tags. Each main set has five n-gram sets, one for each tag type.

The n-grams are gathered from each sentence and they do not cross sentence boundaries. The n-grams do not need to lapse multiple sentences since the system only detects errors from one sentence a time. The n-grams range from length two to five. If a sentence is shorter than an n-gram with a given length, that n-gram is left out. For example, the extraction of word-class-tags for a corpus containing only two sentences, "Koira on iloinen ." and "Se heiluttaa pitkää häntäänsä innokkaasti .", would be conducted as follows:

First, a hashmap data structure is initialized. The hashmap is used to store the n-grams and their frequencies. To start the process, each word in the sentences is replaced with its word class, which is found in the FinnTreeBank corpus in the annotation information of the word. "Koira on iloinen ." becomes "N V A Punct" and "Se heiluttaa pitkää häntäänsä innokkaasti ." becomes "P V A N Adv Punct". Next, the n-gram extraction begins, starting from bigrams working up to fivegrams. Bigrams for the first sentence are (N, V), (V, A) and so on, and for the second sentence (P, V), (V, A) and so on. Each of these n-grams is stored in the hashmap with the n-gram as the key

and its frequency as the value. The frequency value is one when an n-gram is first encountered. Each time an n-gram, that already exists in the hashmap, is encountered, the n-grams frequency value is incremented. Trigrams for the first sentence are (N, V, A) and (V, A, Punct), and for the second sentence (P, V, A), (V, A, N) and so on. Note that the n-grams do not cross sentence boundaries, for example, the bigram (Punct, P) is not included in the final n-gram set. The process continues like this until we reach the extraction of fivegrams. The first sentence contains only four tokens, thus it produces no fivegram. The processing of the second sentence proceeds normally and it produces two fivegrams as it has six tokens.

The above example shows the extraction of word-class-tag n-grams. The same process is conducted for the four other tag types: whole-tags, num/case-tags, num-tags, and case tags. Additionally, for each n-gram set, there is an n-gram set with sentence border tags "START" and "END". This provides the error detection system with additional information on how a sentence can start or end. For example, the word-class-tags of the sentence "Koira on iloinen ." in this format would be: "START, N, V, A, Punct, END". The n-gram extraction process stays exactly the same but the resulting set of n-grams is now slightly different: the bigrams of the sentence are now (START, N) ... (Punct, END), trigrams are (START, N, V) ... (A, Punct, END) and so on. With two tokens added, the sentence now has six tokens, thus fivegrams can also be extracted: (START, N, V, A, Punct) and (N, V, A, Punct, END).

3.4 Statistics

There are a total of 20 reference n-gram sets: FTB1 and FTB3, a set with and a set without border tags for each of them, and a set for each of the five tag types for all four of them. To study the nature of the n-gram sets, frequency distributions and the five most frequent n-grams for each set and for each n-gram length are analyzed. In the frequency distribution tables, the first frequency intervals are 1, 2-5, 6-10 and 11-30, which correspond to the cutoff values that are used in evaluation in section 5.

All of the frequency distribution tables indicate that the higher the n-gram length is, the greater portion of the n-grams is distributed on low frequencies. Naturally, when the n-gram length is higher, there are more possibilities to arrange the components of the n-grams. Table 3 shows the effect for word-class-tag n-grams in FTB1. In the case of bigrams, 16.6% of the n-grams have frequency 10 or lower, whereas, for fivegrams, the percentage is 88.7%.

frequency	bigrams	%	frequency	trigrams	%
1	6	4.3	1	182	16.1
2-5	8	5.8	2-5	248	22.0
6-10	9	6.5	6-10	135	12.0
11-30	17	12.2	11-30	162	14.3
31-2518	85	61.2	31-707	359	31.8
2519-5036	6	4.3	708-1414	31	2.7
5037-7554	4	2.9	1415-2122	4	0.4
7555-10072	2	1.4	2123-2829	6	0.5
10073-12590	2	1.4	2830-3537	2	0.2
total	139	100	total	1129	100
frequency	fourgrams	%	frequency	fivegrams	%
1	1633	30.2	1	7509	44.5
2-5	1572	29.1	2-5	5763	34.2
6-10	651	12.0	6-10	1685	10.0
11-30	791	14.6	11-30	1444	8.6
31-193	675	12.5	31-60	324	1.9
194-386	62	1.1	61-121	113	0.7
387-579	18	0.3	122-182	18	0.1
580-772	4	0.1	183-243	7	0.0
773-965	4	0.1	244-304	3	0.0
total	5410	100	total	16866	100

Table 3: Frequency distribution for word-class-tag n-grams in FTB1

Table 4 displays the effect for whole-tag n-grams in FTB1. Whole-tags carry more information than word-class-tags, and since there are more unique components, there are also more unique n-grams. In this case, the proportion of bigrams with frequency 10 or less is already high, 80.8%. When moving to fivegrams the number is even more dramatic 99.8%.

Tables 3 and 4 also show that when the n-gram length is high, the most frequent n-grams occur less number of times than short n-grams. In word-class-tags, the most frequent bigram occurs 12590 times and the most frequent fivegram occurs 304 times. This, again, happens because bigrams have fewer unique n-grams than fivegrams. In whole-tags, the numbers are lower, because having more unique tag components also means that there are more unique n-grams: the most frequent bigram occurs 5090 times and the most frequent fivegram occurs 45 times. Unsurprisingly, the total number of unique n-grams raises as the n-gram length raises and as we move to tags

that carry more information. These effects are similar for all of the n-gram sets.

frequency	bigrams	%	frequency	trigrams	%
1	2651	37.9	1	19615	61.1
2-5	2293	32.8	2-5	8985	28.0
6-10	707	10.1	6-10	1642	5.1
11-30	715	10.2	11-30	1316	4.1
31-1018	615	8.8	31-189	493	1.5
1019-2036	12	0.2	190-378	32	0.1
2037-3054	5	0.1	379-568	9	0.0
3055-4072	0	0.0	569-757	1	0.0
4073-5090	1	0.0	758-947	1	0.0
total	6999	100	total	32094	100
frequency	fourgrams	%	frequency	fivegrams	%
1	48371	77.6	1	66607	89.0
2-5	11899	19.1	2-5	7769	10.4
6-10	1314	2.1	6-10	325	0.4
11-30	643	1.0	11-30	95	0.1
31-43	71	0.1	31-33	5	0.0
44-86	44	0.1	34-36	1	0.0
87-129	8	0.0	37-40	0	0.0
130-172	2	0.0	41-43	1	0.0
173-216	1	0.0	44-45	1	0.0
total	62353	100	total	74804	100

Table 4: Frequency distribution for whole-tag n-grams in FTB1 for each n-gram length

Although the most frequent n-grams are not too important with regard to the actual grammar checking, because the grammar checker only looks at whether certain n-grams appear in the n-gram set at all, they do provide some information about the nature of the n-gram sets. Table 5 shows the most frequent whole-tag n-grams in FTB1. FTB1 consists of linguistic examples from a descriptive Finnish grammar and it is annotated by hand. The top five n-grams include examples of nouns agreeing with verbs and adjectives agreeing with nouns by their number. Agreement by the grammatical case is also shown, but the examples are not too interesting since almost all the cases are nominative. Punctuation-conjunction-clause structures are also represented. All in all, the examples seem to be proper language, although,

this is a very narrow view of the n-gram set as the vast majority of n-grams is not visible here.

bigram	frequency
N|Sg|Nom, V|Sg3	5090
V|Sg3, Adv	2914
Adv, Pun	2465
Adv, Adv	2294
N|Sg|Nom, Pun	2204
trigram	**frequency**
N|Sg|Nom, V|Sg3, Adv	947
A|Sg|Nom, N|Sg|Nom, Pun	569
Adv, Adv, Pun	541
V|Sg3, Adv, Adv	478
N|Sg|Gen, Adp, Pun	472
fourgram	**frequency**
Pun, CS, N|Sg|Nom, V|Sg3	216
N|Sg|Nom, V|Sg3, Adv, Adv	153
V|Sg3, A|Sg|Nom, N|Sg|Nom, Pun	134
N|Sg|Nom, V|Sg3, Adv, Pun	120
Adv, A|Sg|Nom, N|Sg|Nom, Pun	111
fivegram	**frequency**
V|Sg3, Adv, A|Sg|Nom, N|Sg|Nom, Pun	45
Pun, CS, N|Sg|Nom, V|Sg3, Adv	43
N|Sg|Nom, V|Sg3, Adv, Adv, Pun	37
V|Sg3, Pun, CS, N|Sg|Nom, V|Sg3	37
N|Sg|Nom, V|Sg3, Adv, A|Sg|Nom, Pun	37

Table 5: Five most frequent whole-tag n-grams in FTB1 for each n-gram length

In table 6, the most frequent whole-tag n-grams in FTB3 are displayed. FTB3 is automatically annotated and collected from various Finnish corpora, so it could contain more "rubbish" data or incorrect examples. The only tags that are represented in the table are Pun, Num|Sg|Nom, N|Sg|Nom and N. An n-gram, where all the elements are Pun is included in the five most frequent n-grams for all n-gram lengths. The other n-grams are alternation of punctuation and some other tag. These are caused by lists of items or numbers separated by commas, and dates appearing in the data. This indicates that the FTB3 n-gram sets contain more ungrammatical n-grams, which means

that the grammar checker based on FTB3 is likely to pass ungrammatical sentences as correct ones. This does not affect the error detector's capability of passing grammatical sentences as correct, assuming that correct examples are represented elsewhere in the data.

bigram	frequency
Num|Sg|Nom, Pun	2392975
N|Sg|Nom, Pun	2209140
Pun, Num|Sg|Nom	1652323
Pun, N|Sg|Nom	1370185
Pun, Pun	1204254
trigram	**frequency**
Pun, Num|Sg|Nom, Pun	1290344
Num|Sg|Nom, Pun, Num|Sg|Nom	932582
Pun, N|Sg|Nom, Pun	784968
Pun, Pun, Pun	420598
N|Sg|Nom, Pun, N|Sg|Nom	403842
fourgram	**frequency**
Num|Sg|Nom, Pun, Num|Sg|Nom, Pun	716259
Pun, Num|Sg|Nom, Pun, Num|Sg|Nom	510204
N|Sg|Nom, Pun, N|Sg|Nom, Pun	238666
Pun, Pun, Pun, Pun	231253
Pun, Num|Sg|Nom, Pun, N|Sg|Nom	209534
fivegram	**frequency**
Pun, Num|Sg|Nom, Pun, Num|Sg|Nom, Pun	451486
Num|Sg|Nom, Pun, Num|Sg|Nom, Pun, Num|Sg|Nom	333327
Pun, Pun, Pun, Pun, Pun	169293
Num|Sg|Nom, Pun, Num|Sg|Nom, Pun, N|Sg|Nom	150138
Pun, N, Num|Sg|Nom, Pun, Num|Sg|Nom	127116

Table 6: Five most frequent wholeTag-tag n-grams in FTB3 for each n-gram length

Table 7 shows five randomly selected fivegrams from FTB3. The n-grams have structures that seem to be more representative of natural language than the five most frequent n-grams, which suggests that the FTB3 n-gram set does indeed contain grammatically correct examples. Additionally, having obscure n-grams is beneficial if one tries to avoid over flagging of rare correct structures as ungrammatical, but this comes with the expense of the system also passing some ungrammatical structures as correct.

fivegram	frequency
N|Sg|All, V|Sg3, V|Sg|Nom, A|Sg|Ill, N|Sg|Ill	2
Pun, N|Pl|Par, V, V, V|Abe	2
V|Sg3, V|Sg|Nom, Adv, N|Pl|All, A|Pl|Ess	2
V|Pl|Par, N|Pl|Par, N|Pl|Gen, N|Sg|Ill, V|Pl|Ess	1
V|Pl|Nom, A|Pl|Nom, N|Pl|Nom, A, N|Pl|Ess	1

Table 7: Random sample of whole-tag fivegrams in FTB3

4 Method

The grammar error detection method in the work of this thesis is based on the paper by Nazar and Renau (2012), who check Spanish grammar by using Google Books N-gram Corpus: if the n-grams of an input sentence appear in the Google n-grams, the sentence is grammatically correct, otherwise not. Finnish is a language with complex morphology and extensive generative properties (Karlsson, 2008), thus plain word n-grams are not used in this case. Instead, the error detector in my work utilizes POS n-grams, which is similar to the approach of Alam et al. (2007), who check Bangla grammar with POS n-grams. Using POS n-grams allows us to have a general picture of a grammar without the set of reference n-grams growing to an unrealistic size. The grammar checking system has three main phases when detecting errors from an input sentence:

First, conduct morphological analysis for each word in the sentence and produce POS-tags based on the analysis. Each word might have multiple different interpretations.

Next, select which morphological interpretations of the sentence will be passed forward in the system.

Finally, see whether the POS n-grams of the sentence are found in the reference set of n-grams.

4.1 Morphological analysis

The tokens of an input sentence are morphologically analyzed with Omorfi (Pirinen, 2015). Omorfi gives a list of all possible analysis interpretations of a token. Omorfi provides extensive information about the morphology of a word but we are only interested in the word class, number, and case of each

token, as the checker mainly aims to detect errors related to number and case agreement. For each item in the list of interpretations, word class, number and case are fetched, and the information from these are converted to suit the format in the n-gram sets collected from FTB. Usually uppercasing the first letter and lowercasing the rest of the letters is enough, but in some cases, the tag has to be further altered. Additionally, proper nouns are considered to be nouns, and auxiliary verbs are considered to be verbs, as they are in the reference n-gram sets.

4.2 Sentence interpretation selection

Each word might have multiple different interpretations. For example, the word "multa" could mean "soil" (N Nom Sg) or "from me" (Pron Abl Sg1). Since each word can have multiple interpretations, the whole sentence could be interpreted in multiple different ways. Two different methods of formulating an interpretation for a sentence are experimented with in this work. The first method is to choose the most likely interpretation for each token based on simple statistics from FinnTreeBank 1. The second method is to use brute force and evaluate all possible sentence interpretations based on the list of tags for each token.

4.2.1 Check the most probable interpretation

The first method is to check the most probable (CMP) interpretation based on statistics from FTB1. To setup this method, a dictionary of word-class frequencies is gathered from FTB1. The dictionary contains an entry for each word that appears in FTB1 and the number of times the word represents each word class. In the entries, the words are first converted into their base forms. For example, the entry for "olla" is {V: 8853, N: 26, Adv: 25, A: 5}, which indicates that "olla" is most often a verb. The dictionary can then be used to select an interpretation for each word of an input sentence based on the list of analyses from Omorfi. The base form of the first word item of the Omorfi list is selected and its word class is chosen to be the word class that has the highest frequency value in the word's entry in the dictionary. If the word is not contained in the interpretation dictionary, the word class in the first analysis of the Omorfi list is used.

Table 8 shows an example of choosing the interpretation for the sentence "Multa tuli maasta .". First, each word is analyzed with Omorfi, which provides the possible base forms and their word classes. The first base form

in the analysis list is selected. In this case, the base forms are “multa”, “tuli”, “maa” and “.”. Notice that this phase completely ignores the possibility for the base form of “multa” to be “mä”, which is one of the options provided by Omorfi. Next, the base forms are used to find entries from the word class frequency dictionary. From each entry, the most frequent word class is chosen. The most frequent word classes in FTB1 for these base forms are “N”, “V”, “N” and “Punct”. Finally, for each word, the analysis that contains the chosen word class is selected from the list from Omorfi, which results in the final interpretation “N Nom Sg, V Sg3, N Ela Sg, Punct”.

input sentence	“Multa”	“tuli”	“maasta”	“.”
convert to base form	“multa”	“tulla”	“maa”	“.”
most frequent word class	N	V	N	Punct
interpretation from Omorfi	N Nom Sg	V Sg3	N Ela Sg	Punct

Table 8: Example of choosing the most probable interpretation for “Multa tuli maasta.”

If the list of analyses from Omorfi contains multiple analyses with the most frequent word class, the method chooses the last interpretation with the correct word class. For example, if the word is “alusta” with the correct word class being “N” and the list of analyses is {“N Nom Sg”, “N Par Sg”, “N Ela Sg”}, the last one of these “N Ela Sg” will be selected.

This method is very simple to implement but it has major shortcomings. The choosing of the base form is quite arbitrary as only the first option in the list of analysis from Omorfi is considered. Even in a case where the base form is selected correctly, but the entry in the word class dictionary contains a very even distribution, for example, {V: 501, N: 499}, the most frequent option, “V”, is always chosen, even though “N” would be the correct option in almost half of the cases. Thus, this method is likely to pass an incorrect interpretation forward to the grammar checking phase, which would result in grammatically correct sentences being flagged as ungrammatical. This method is expected to be outperformed by the all possible interpretations method, which is described in section 4.2.2.

4.2.2 Check all possible interpretations

The second interpretation selection method checks all possible (CAP) interpretations of the input sentence. This covers the possibility that sentences might have multiple different morphological interpretations. This brute force natured method creates all possible interpretations for the input sentence based on the list of analyses for each token from Omorfi. Each of the interpretations is sent forward to the grammar error detection phase, and if any of them passes the checking, the input sentence is considered to be grammatically correct.

Let's say we process the sentence "Multa tuli maasta ." with this method. First, each of the tokens is analyzed with Omorfi. The words "multa" and "tuli" both have a list with two different interpretations, and "maasta" and "." both have a list with one interpretation. Next, all possible interpretations, even nonsensical ones, are formed based on the tags, as shown in table 9. In this case, the sentence has four different interpretations, each of which is checked for correct grammar one by one. Let's assume that the two first interpretations do not pass the checking phase, but the third one does. The whole sentence is then determined to be grammatically correct, and the fourth interpretation does not need to be sent to the grammar checking phase.

"Multa"	"tuli"	"maasta"	"."
Pron Abl Sg1	V Sg3	N Ela Sg	Punct
Pron Abl Sg1	N Nom Sg	N Ela Sg	Punct
N Nom Sg	V Sg3	N Ela Sg	Punct
N Nom Sg	N Nom Sg	N Ela Sg	Punct

Table 9: Example interpretations of sentence "Multa tuli maasta."

If the input sentence is long and it has ambiguous tokens with multiple different interpretations, the processing can take a large amount of time. For example, the word "voi" has 8 unique interpretations in Omorfi. In an extreme case, the input sentence might be the word "voi" ten times. The number of interpretations would be $8^{10} = 1073741824$, which is unreasonable to start to process. However, there is a backup method. If the formulation of all interpretations takes more than one second, the processing is interrupted and a backup method is activated. The original input sentence is now split into fivegrams, which are then checked separately. For example, the sentence (with indexes for clarification) "voi^1 voi^2 voi^3 voi^4 voi^5 voi^6 voi^7 voi^8 voi^9

voi^{10}" would be split to six new sentences:

"voi^1 voi^2 voi^3 voi^4 voi^5"
"voi^2 voi^3 voi^4 voi^5 voi^6"
"voi^3 voi^4 voi^5 voi^6 voi^7"
"voi^4 voi^5 voi^6 voi^7 voi^8"
"voi^5 voi^6 voi^7 voi^8 voi^9"
"voi^6 voi^7 voi^8 voi^9 voi^{10}"

Each of these sentences is sent back and processed again with the all possible interpretations method. If all of the sentences pass the process, i.e. if all of the sentences have at least one interpretation that passes the grammar error detection phase, the original input sentence is considered to be grammatically correct. The intuition is, that since the fivegrams are the longest n-grams that are looked at in the grammar detection phase, and since the fivegrams of the split sentence overlap with each other, the grammar checker should be able to detect more or less the same errors in a split sentence, as it would in a non-split sentence. Now, the total number of interpretations is reduced to $6 * 8^5 = 196608$, which means that there are approximately 5000 times fewer options to be checked.

This method is still quite simple to implement, but it requires more processing time than the previously described most probable interpretation method, even with the attempt to reduce the amount of computing with the backup method. However, the all possible interpretations method is more likely to find the correct interpretation of the sentence. It is also more likely to pass sentences as grammatically correct in general, because it sends even obscure interpretations to the grammar detection phase. This might not be undesired behavior, as a grammar checker, that flags errors in sentences that in reality have no errors, seems more useless, than a grammar checker that some times leaves grammatically incorrect sentences unflagged, but every time a sentence is flagged, the sentence does actually contain grammar errors.

4.3 Grammatical error detection

Once a morphological analysis is received from the interpretation formulation phase, the POS n-grams from the input sentence are compared to the set of grammatically correct POS n-grams collected from FinnTreeBank. If all POS n-grams from the input sentence are found in the set of correct POS n-grams, the input sentence is considered to be grammatically correct. A cutoff value

can also be assigned to the grammar checker. This value sets a frequency threshold that the n-grams have to cross in order to be considered correct. The cutoff value is compared to the frequency value that is paired with each n-gram in the reference n-gram sets.

The POS n-gram checker determines the correctness of the input sentence by checking whether each of its n-grams is presented in the set of correct POS n-grams enough number of times based on the cutoff value. First, all n-grams from length two to five (in that order) are checked starting from the first word. If no errors are found, the algorithm moves to the next word and conducts the process starting from the second word. This procedure repeats until an error is found, or until all n-grams are checked. If an error is found, i.e. an n-gram is not found in the set of correct n-grams with a sufficient frequency value, the iteration stops, and the location of the erroneous n-gram is returned. Because the processing ends at the first error found, the system only reports the first grammar error in the sentence. The location of the error is a pair of values which represent the positions of the first and the last POS-tags of the error n-gram in the sentence. When the error detection is done using the most probable interpretation method, the location of the erroneous n-gram is directly the location of the error in the sentence, because there is only one interpretation that needs to be checked. However, when using all interpretations method, the location of the error can only be estimated, because the input sentence is considered to be erroneous only if an error is found in all of its interpretations. In this case, the errors in each interpretation are not likely to have the exact same location value, thus it is not clear what is the exact location of the error in the sentence as a whole. To estimate where the error is, the location is chosen to be one that is represented the most times in the interpretations. For each interpretation, the location of the erroneous n-gram is recorded in a list. After processing all the interpretations, the location that occurs the most times in the list is reported to be the location of the error in the input sentence. If there are multiple location values that appear the same number of times and more times than the others, the first one of those values is chosen. The error location selection for the all interpretations method is somewhat arbitrary and the check all possible interpretations method is not able to exactly locate the error. However, it is able to determine if there is an error somewhere in the sentence, and whether the sentence as a whole is grammatically correct or not.

5 Evaluation

In the evaluation phase, the aim is to assess how using different reference n-gram sets, error detection methods, cutoff points and tag types in the grammar error detection system perform against one another. The test data used in the evaluation is gathered from machine-translated sentences. Each grammar error in the data set is labeled with an error type. The data is further described in section 5.1. The evaluation is conducted using multiple different grammar checker setups in multiple different settings. This is explained in section 5.2.

5.1 Test data

A test data set, that consists of Finnish sentences with grammar errors marked, is required to evaluate the grammar error detection system. Additionally, each error in the data set should be labeled with an error type, which allows the data to be used to measure error type detection performance. Such a data set seems to not exist for Finnish, thus one is created within the scope of this thesis.

Ideally, the test data would consist of authentic text in order to assess how the system performs in a real-world setting, as suggested by Tschichold (1994). However, as such data is not found for Finnish, the test data for evaluation is constructed using Finnish text produced by a machine translation system. More specifically, data from Helsinki phrase-based English to Finnish machine translator[4] submitted to Statmt (Koehn, 2005) newstest2015 is used. Using sentences produced by a phrase-based statistical translator is beneficial because it is likely to produce actual grammatical errors, where language structures are clearly ungrammatical. The following translations from the 2015 statistical phrase-based translator have clear examples of incorrect grammar; the first sentence has a verb in infinitive form when it should be a participle, and in the second one, an adjective is singular nominative and it modifies a noun that is plural partitive:

Tuntuu, että kaikki on hakkeroida, hän kirjoittaa mukaan.

Fiksu tapoja säästää oppikirjoja

[4] `http://matrix.statmt.org/systems/show/2498`

In contrast, neural network based machine translation systems are often able to produce proper grammar while the translation might have other issues. The following examples from Helsinki English to Finnish neural machine translation submission for newstest2018[5] are grammatically correct but have unusual semantics:

Patsaan merkitys on täysin käsitys kulmasta.

Katossa oleva ämpäri toimii latteana.

Additionally, observing the sentences produced by the two translation systems mentioned above shows that the phrase-based translator creates more errors in general, which is helpful when creating a grammar error data set.

The data from Statmt newstest2015 Helsinki phrase-based system for English to Finnish consists of English input sentences, Finnish output translations, and Finnish reference sentences, that represent the correct translations. The Finnish output translations are likely to contain grammar errors, thus 764 sentences, that are manually verified to contain grammar errors, are selected from this set to represent grammar error examples in the test data. An error can be located in one word, or it can span multiple words or an entire phrase. Additionally, each error has been labeled with an error type by hand. The error types are described in section 5.1.1. An example of an entry in the test data might look like this:

Juankoski liitetään Kuopion kaupungin vuoden 2017 alussa .
 # ((wrong case)) # ((3, 3))

In the above sentence, the error can be pinpointed to the fourth word. The word is in the genitive case while it should be in illative in order to be grammatically correct. The error type label is placed after the example sentence. The location information is the last item in the entry. It contains two values which indicate the locations of the first token and the last tokens of the error.

In order to have grammatically correct examples in the test data, 764 sentences are chosen from the set of Finnish reference sentences. Each of these sentences is a correct reference sentence for one of the 764 previously selected grammar error examples. In its entirety, the test data consists of 1528 sen-

[5] `http://matrix.statmt.org/systems/show/3360`

tences: 764 sentences with grammar errors marked and labeled with an error type, and a grammatically correct version for each of these sentences.

There is also a second version of the test data, where foreign proper names, that are not recognized by Omorfi (Pirinen, 2015), are replaced by Finnish names to decrease the number of unknown words. The no-unknown-names (NUN) version of the test data was created by checking each word in the data with Omorfi. If the word is labeled "unknown" by Omorfi, the word is further manually evaluated. If the word is recognized to be a foreign name, it is replaced by a Finnish name. This procedure is done for each sentence in the test data, and the resulting sentences are stored in the NUN version of the data. The original test data has 1137 words unrecognized by Omorfi, while the same number for the NUN version is 346. The NUN test data was created because having a test set full of words unknown to the morphological analyzer hinders the grammar checkers overall performance. In a real-world setting, the checker should, of course, be able to handle any kind of foreign name usage, but at this stage, it is interesting to see how the grammar checker would perform if it had a robust named entity recognition system in its pipeline.

The NUN test data has the exact same number of errors and exactly the same error types as the original test data set. This is ensured by the fact that the error marking and labeling as well as the foreign name replacements are both conducted manually. If a name is replaced outside a grammar error, there are naturally no new errors introduced into the sentence. If the replacement happens within an error that is already labeled, the location of the error and the error type are preserved. If a word is labeled as an error in the manual error labeling phase because it is an unknown word, there is no possibility for it to be replaced by a Finnish name in the name replacement phase, because the manual interpretation of it not being a foreign name is same in both phases. When checking the sentences in the test data, the fact that foreign names are replaced allows Omorfi to recognize more tokens, and fewer "unknown" tags are sent to the grammar error detection process.

5.1.1 Error types

To evaluate what kind of grammar errors each checker setup is able to detect, the errors in the test sentences are labeled with an error type, which is selected from 12 options. The error types and the number of their occurrences in the test data are shown in table 10.

Error type	Occurrences
extra structure	34
foreign word	48
missing predicate	90
missing structure	42
unknown word	57
wrong case	191
wrong case and number	8
wrong number	16
wrong person	14
wrong structure	253
wrong word class	8
wrong word order	3

Table 10: Distribution of error types

The error types were chosen while examining the grammar errors in the translations from the phrase-based machine translation system. The selected error types are the most distinct ones appearing in the translations. The types are described using examples with annotations containing case and number information in the following:

Extra structure:

An error labeled with an extra structure adds invalid structure to the sentence, which causes grammatical errors. The added structure can be punctuation, a word or multiple words.

kuva	**,**	**ja**	**on**	**sittemmin**	**poistettu**
picture +sg+nom	extra-comma	extra-conjunction	**be**+sg3	**since**	**remove** +sg

the picture extra-comma extra-conjunction **has since been removed**

Foreign word:

Foreign word indicates that a sentence contains a word that is identified by the author to be in a different language than Finnish.

Vielä	appetite	for	koteja	on	ollut	siunaus ...
Still	foreign	foreign	**home** +pl+par	**be**+3sg	**be**+sg	**blessing**

Still foreign foreign **homes has been a blessing...**

Missing predicate:

Missing predicate means that the sentence is missing a verb that acts as the predicate, which makes the example ungrammatical.

Luettelon	pikahuutokaupasta	ensi	viikolla
Catalogue+gen	**fast-auction**+sg+ela	**next**	**week**+sg+ade

A catalogue from a fast auction next week

Missing structure:

Missing structure indicates that something essential for the sentence's grammatical correctness is missing. The missing part can be for example an adposition, a conjunction, a subject or an object or it can span multiple words. Missing predicate has its own label, because it is a more common error than other missing components.

The example sentence is missing a conjunction or a sentence boundary between "alussa" and "tapahtuma".

... joka	on	juuri	saa-punut	alussa	tapah-tuma	voi	olla
which	**be** +sg3	**just**	**arrive** +sg	**begin-ning** +sg+ine	**event** +sg+nom	**can** +sg3	**be**

... which has just arrived in the beginning the event can be ...

Unknown word:

Unknown words are tokens that are not recognized by the author to be Finnish or any other language. There is a possibility that some errors that should be labeled as foreign words were instead labeled as unknown because the language was not recognized.

Eli	**vitsaiiia**	**usein**	**hänet**	**vielä**	**vaikeuksiin**
So	unknown	**often**	**he**+acc	**still**	**trouble**+pl+ill

So unknown **often him still into trouble**

Wrong case or wrong number:

Wrong case or wrong number means that a word has the wrong grammatical case or number in relation to the surrounding words. A specific case and number are required to satisfy word agreement. Additionally, certain structures, e.g. adpositions, might require a certain case or number. There are three labels related to these errors: **wrong case**, **wrong number** and **wrong case and number**.

In this example sentence, "lisäkaudet" should be singular and in adessive, "lisäkaudelle", and the error would be labeled as wrong case and number.

Ehdo-tukseen	**sisältyy**	**myös**	**optio**	**kahdelle**	**lisäkaudet**
proposal +sg+ill	**include** +sg3	**also**	**option** +sg+nom	**two**+sg+ade	**additional-season** +pl+nom

The proposal also includes an option for two additional seasons

Wrong person:

Wrong person is closely related to wrong number. The difference is that grammatical number refers to nominals, which can be either singular or plural, while grammatical person refers to verbs, which, in addition to being either singular or plural, are also in first, second or third person.

In this example, "jotka" is plural, and "kommentoi" should be in third plural, "kommentoivat", instead of third singular.

jotka	**kommentoi**
who+pl	**comment**+sg3

who commentates

Wrong structure:

Wrong structure means that a linguistic structure is erroneous, but it is very difficult to pinpoint the exact reason why. The structure is usually completely incoherent and contains multiple errors that are difficult to separate.

This example has multiple errors, and it is difficult to find the exact location of the error. "vetooikeuksia" is missing a hyphen between the two "o"s and its conjugation does not match to the surroundings. "vahtikoiraryhmä" is in wrong grammatical case in relation to its surroundings. "teki virallisen valituksen vasemmalle" is a grammatically correct structure, but it does not match to its surrounding. The sentence requires multiple large edits to make it grammatical.

Että	**vetooi-keuksia**	**teki**	**virallisen**	**valituksen**	**vasem-malle**	**vahtikoira-ryhmä**
that	**veto** +pl+par	**make** +sg3	**official** +sg+gen	**complaint** +sg+gen	**left**	**watch-dog-group** +sg+nom

That at vetos made an official complaint to the left watch dog group

Wrong word class:

Wrong word class means that a word has the correct root, but it is derived to be in the wrong class.

In the example, the adverb "humalassa" should be replaced with the adjective "humalainen"

Olen	**humalassa**	**demokraatti**	**Texasissa**
be+1sg	**drunk**	**democrate**+sg+ine	**Texas**+sg+ine

I am drunk democrate in Texas

Wrong word order:

Wrong word order means that the words being in the wrong order makes the sentence erroneous. Even though the word order in Finnish is quite free compared to many other languages, there are cases where the wrong order can create an error.

In the following example, "Klubin" should be placed before "pelaaja". A structure where something, that is owned, is placed before the owner is very rarely used and it is considered to be an error.

mutta	**tilalle**	**tuli**	**pelaaja**	**Klubin**
but	**to-replace**	**come**+sg3	**player**+sg+nom	**Klubi**+sg+nom

but to replace him came a player Klubi's

5.2 Test settings

The evaluation is conducted to multiple grammar checker setups based on different selections of reference n-gram sets, error detection methods, cutoff points, and tag types. There is a total of 200 setups that combine the aforementioned features as shown in table 11. Precision, recall, and f-score values are used to evaluate each of the test setups, as per usual when evaluating grammar checkers (Lin et al., 2011; Nazar and Renau, 2012; Silva and Finger, 2013). Recall is the number of correctly flagged errors divided by the total number of errors in the test data. Precision is the number of correctly flagged errors divided by the number of correctly and incorrectly flagged errors. F-score is calculated as $2/(precision^{-1} + recall^{-1})$. Precision and recall provide interesting information on how the grammar checker is tuned to behave. Usually, a grammar checker with high precision is favored over a checker with high recall. Tschichold (1994) and Tschichold et al. (1997) state that an overflagging grammar checker is confusing and misleading, especially for non-native speakers. A user satisfaction evaluation conducted by Deksne and Skadiņš (2011) also shows, that users generally prefer checkers with high precision rather than high recall.

There are five comparisons that are made by measuring precision, recall and f-score values:

- comparing FTB1 and FTB3
- comparing WB and WoB
- comparing CMP and CAP
- comparing cutoff thresholds 0, 1, 5, 10 and 30
- comparing different POS-tag n-grams

These comparisons are conducted using the average scores across all grammar checker setups that use a specific feature. For example, when examining the precision between FTB1 and FTB3, the average precision value of all setups that use FTB1 is compared to the average precision value of all setups that use FTB3. The cutoff point selections (0, 1, 5, 10 and 30) were guided by the n-gram set statistics which were described previously in section 3.4. The thresholds were chosen to be very low as a low value already removes a significant portion of the reference n-grams. For example, cutting of the n-grams that appear only once in the FTB1 whole-tag n-gram reference set removes 89% of the reference n-grams in that set.

Data set	Border tags	Method	Cutoff point	POS-tag
FTB1 FTB3	WB WoB	CMP CAP	0 1 5 10 30	Word-class-tag Whole-tag Number/case Number Case

Table 11: Different test setups. Each of the setups uses one option from each category. In total, there are 200 different grammar checker setups.

Further, the grammar checker setups are evaluated by their ability to detect different error types. This measure is taken by calculating the proportion of each error type labeled correctly by the system. Special focus is put on analyzing how different POS-tag based reference n-gram sets perform in this regard. The measurements are, again, reported as the average error type detection proportion values of all setups that use a specific POS-tag.

Each of the 200 setups is run through the two versions of the test data: the original test set and the NUN version of the test set. Additionally, each setup is evaluated in two different settings: setting 1, where the grammar checker has to flag the location of the error in a sentence, and setting 2, a relaxed setting, where the checker only has to decide whether the whole sentence is grammatically correct. For each test set and setting, the grammar checker setups that have the best precision, recall, f-score, and error type detection accuracy are reported.

In setting 1, if the actual location of the error overlaps even partially with the location reported by the checker, the location is considered to be correct. Although ideally, it would be desirable for the location detection to be perfect, expecting the current version of the grammar checker system to report the

locations exactly correctly would be very strict. The error locations in the test set are marked by hand and it would be unreasonable to think that the error locations emerging from the reference n-gram sets would have the exact same intuition as a human.

Each example error sentence can contain multiple grammar errors, and the total number of individual grammar errors in the test data sums up to 1226. However, as the grammar checker is only able to detect the first error of a sentence, only the total number of first errors, which is the same as the number of error example sentences: 764, is used in recall calculations.

6 Results and analysis

In this section, the results of the evaluation are presented and analyzed. The precision, recall and f-score performances are reported in section 6.1. First, the average values for all the test setups in each of the four settings are reported. The four test settings are normal test set in setting 1, normal test set in setting 2, NUN test set in setting 1 and NUN test set in setting 2, where normal test set is the original set, NUN set is the test set with unknown foreign proper names removed, setting 1 requires detection of the location of the error and setting 2 only requires the whole sentence to be flagged as erroneous. Next, the average values are reported in all four test settings for each of the following comparisons: FTB1 against FTB3, WB against WoB, CMP against CAP, different cutoff points against each other, and different POS-tags against each other. Finally, the individual grammar checker setups, that performed the best with regard to precision, recall, and f-score, are reported.

The error type detection performances are reported in section 6.2. The proportions of correctly labeled errors are measured for both test sets, but only in setting 1, as it is not meaningful to consider the checker to detect error types correctly when whole sentences are marked erroneous. The performance comparison is reported only for POS-tags, as the other comparisons yield uninteresting results. Further, the best individual grammar checker setups in detecting each error type are reported.

6.1 Precision, recall and f-score

Table 12 shows the average precision, recall, and f-score of all the 200 grammar checker setups in all four settings. In setting 1, where the checker has

to locate the error in a sentence, the average performance is very poor with regard to both precision and recall. The checker finds few errors from the total set, and even when it does flag an error, the flagged item is not actually an error in most cases. The reason, why the error detection performs poorly, is that most of the checker setups are very strict and they falsely flag errors right at the start of most sentences, which causes the checker to not reach the locations of the true errors. The restrictiveness of the grammar detection setups is explained further in the following comparison sections.

	Setting 1 Normal test set	Setting 2 Normal test set	Setting 1 NUN test set	Setting 2 NUN test set
precision	0.241	0.538	0.307	0.618
recall	0.317	0.718	0.286	0.609
f-score	0.267	0.599	0.266	0.552

Table 12: Overall average precision, recall and f-score

All three values are higher in setting 2, where the checker marks whole sentences as erroneous, compared to setting 1. This was to be expected as the error detection system has a better chance of finding an error in setting 2. However, it is trivial to achieve a high recall score in setting 2. If the checker is very strict and it flags every sentence as erroneous, the recall value raises to 1.000. Additionally, in this data set where half of the sentences contain errors, if all sentences are flagged as erroneous, a precision score of 0.500 is automatically achieved.

In NUN test set, where foreign proper names have been manually replaced with Finnish names, the average precision is higher and the recall is lower than in the original test set. The checker is less likely to find unknown words in the morphological analysis phase, and proper POS n-grams are sent forward to the error detection phase. POS n-grams, that contain no unknown tags, have a better chance of being represented in the reference n-gram set than n-grams, that do contain them. This explains why the precision is higher, as the checker will not flag structures where the error would be caused by a foreign name being an unknown word. This is also the reason why fewer errors are flagged in general and the recall is lower. The overall performance is poorer in NUN test set compared to the original test set according to the average f-score values.

6.1.1 FTB1 vs FTB3

The test setups that use FTB1 as their reference n-gram set are compared to the setups that use FTB3. FTB1 consists of 19000 sentences that are linguistic examples from Finnish grammar and it is manually annotated. FTB3 contains 4367000 sentences, which are collected from publicly available Finnish corpora and it is automatically annotated.

	Setting 1	FTB1	FTB3	**Setting 2**	FTB1	FTB3
Normal test set	precision	0.232	**0.251**	precision	0.520	**0.557**
	recall	**0.352**	0.281	recall	**0.802**	0.634
	f-score	**0.275**	0.258	f-score	**0.620**	0.578

	Setting 1	FTB1	FTB3	**Setting 2**	FTB1	FTB3
NUN test set	precision	0.261	**0.351**	precision	0.548	**0.687**
	recall	**0.337**	0.235	recall	**0.729**	0.488
	f-score	**0.279**	0.252	f-score	**0.592**	0.511

Table 13: The average scores of test setups using FTB1 and FTB3

FTB1 has higher recall and lower precision than FTB3 in both settings and both data sets as seen in table 13. Even though FTB1 consists of a variety of linguistic examples, it is a much smaller corpus, and it seems to be more restrictive than the much larger FTB3. FTB3 is likely to contain more rarely used or obscure language and, for example, idioms that might require specific grammatical cases. Thus, when using FTB1, more language structures are flagged as errors even though they might actually be just rare phrases, which is reflected in the precision and recall scores. Recall is higher since more sentences are flagged as erroneous in general, but simultaneously this hinders precision. However, the FTB1 setups perform better than FTB3 when taking into account both precision and recall according to the f-score value.

6.1.2 WB vs WoB

The test setups that use n-gram sets with sentence border tags added (WB) are compared with the setups that do not use border tags (WoB). When the grammar checker uses a set with border tags, the sentence boundaries are also marked in the input sentence before following through with the grammar error detection process. Utilizing border tags provides the system with information on how phrases can start and end.

	Setting 1	WB	WoB	**Setting 2**	WB	WoB
Normal test set	precision	0.211	**0.271**	precision	0.538	**0.540**
	recall	0.287	**0.346**	recall	**0.723**	0.712
	f-score	0.237	**0.296**	f-score	**0.600**	0.597
	Setting 1	WB	WoB	**Setting 2**	WB	WoB
NUN test set	precision	0.246	**0.369**	precision	0.615	**0.621**
	recall	0.254	**0.318**	recall	**0.616**	0.602
	f-score	0.226	**0.305**	f-score	**0.554**	0.549

Table 14: The average scores of test setups using WB and WoB

In setting 1, not using sentence borders outperforms using them in both precision and recall, as shown in table 14. When border tags are utilized, the reference POS n-gram set contains more unique n-grams: all the n-grams of the set without borders and additional n-grams of length two to five, which contain the sentence start and end border tags, for each sentence in FTB. In a sense, the WB version conducts the grammar checking for all the same n-grams as WoB, but it runs an additional error detection process for the beginnings and endings of each test sentence. Thus, WB flags errors more in the beginnings of sentences where the process stops, while the actual errors might be somewhere further down the sentence, which causes incorrect error location detection leading to low recall and precision.

In setting 2, the differences between WB and WoB are minuscule, but WoB does have slightly higher precision and WB has slightly higher recall and f-score. The error location issue described above does not have as strong of an effect in this setting, where whole sentences are flagged erroneous: in cases where a WB system determines the error to be at the border of a sentence, but the actual error is somewhere in the middle, the system is still considered to the detect the error correctly, because the error is found in an erroneous sentence. WB having higher recall than WoB is caused by WB being slightly more restrictive and flagging errors in a handful more sentences. However, WB has lower precision, because some of those additional flagged sentences are not always actually erroneous.

6.1.3 CMP vs CAP

All the error detection systems, that choose the most probable (CMP) interpretation of an input sentence, are compared to the systems that choose all

possible (CAP) interpretations. The CMP interpretation selection method is based on naive statistics, while CAP exhaustively checks all different combinations of an input sentence's POS-tags, even obscure ones.

	Setting 1	CMP	CAP	**Setting 2**	CMP	CAP
Normal test set	precision	0.239	**0.243**	precision	0.532	**0.546**
	recall	**0.345**	0.288	recall	**0.777**	0.658
	f-score	**0.276**	0.256	f-score	**0.618**	0.580

	Setting 1	CMP	CAP	**Setting 2**	CMP	CAP
NUN test set	precision	0.289	**0.326**	precision	0.589	**0.647**
	recall	**0.321**	0.250	recall	**0.686**	0.532
	f-score	**0.280**	0.251	f-score	**0.585**	0.518

Table 15: The average scores of test setups using CMP and CAP

Table 15 shows that CMP has higher recall and f-score but lower precision than CAP in both settings and both test sets. CAP tries to detect errors in all possible interpretations of the input sentence, and if any one of them does not contain errors, the original input sentence is considered to be grammatically correct. CMP is much more restrictive, as it chooses only one interpretation to be checked for errors, and the interpretation itself has the possibility of being wrong. Thus, CMP flags more errors in general leading to higher recall. CAP allows obscure, and sometimes even wrong, interpretation to pass the error detection and it also passes grammatical structures as correct more often than CMP, which causes higher precision.

In normal test set, the precision is only slightly higher for CAP. Normal test set contains a higher amount of unknown words because it includes foreign proper names that Omorfi (Pirinen, 2015) cannot interpret. Both CMP and CAP interpretation selection methods are based on Omorfi, which tags the foreign names always as "unknown". As a result, the interpretation produced by CMP and all the interpretations produced by CAP will contain an "unknown" tag in the same position. POS n-grams that contain "unknown" tags are not represented in the reference n-gram sets. Thus, both CMP and CAP will flag an error where the "unknown" tag is, both in the same location, leading to similar precision scores. In NUN test set, the foreign names are replaced by Finnish ones and they are not flagged as "unknown", which allows the checkers to move on to find other errors, which in turn creates more of a difference in precision between normal and NUN test sets. The recall

difference is similar in both normal and NUN test sets as CMP flags more errors in general outside of the foreign name cases.

6.1.4 Cutoffs

The checker is evaluated using different n-gram frequency cutoff values. In the grammar checking phase, if an n-gram from the input sentence is found in the reference n-gram set, but the reference n-gram's frequency value is not higher or equal to the threshold specified in the cutoff value, the input n-gram is considered ungrammatical. The cutoff points used in the evaluation are 0, 1, 5, 10 and 30. These values were chosen based on statistics displayed in section 3.4.

	Setting 1	0	1	5	10	30
Normal test set	precision	**0.249**	0.247	0.242	0.237	0.232
	recall	0.303	0.301	0.318	0.324	**0.337**
	f-score	0.266	0.265	**0.267**	**0.267**	**0.267**

	Setting 2	0	1	5	10	30
Normal test set	precision	**0.547**	**0.547**	0.538	0.534	0.527
	recall	0.672	0.672	0.720	0.742	**0.782**
	f-score	0.587	0.587	0.600	0.605	**0.615**

	Setting 1	0	1	5	10	30
NUN test set	precision	**0.333**	0.330	0.306	0.292	0.277
	recall	0.265	0.263	0.288	0.297	**0.317**
	f-score	0.264	0.262	0.266	0.266	**0.269**

	Setting 2	0	1	5	10	30
NUN test set	precision	**0.649**	**0.649**	0.615	0.599	0.578
	recall	0.544	0.544	0.612	0.644	**0.700**
	f-score	0.531	0.531	0.553	0.563	**0.580**

Table 16: The average scores of test setups using cutoff values 0, 1, 5, 10 and 30

Table 16 shows the results for all cutoff points. In both test sets and both settings, the precision is highest on low cutoff values, and recall and f-score are highest on high cutoff values. When the cutoff value is high, n-grams with low frequency values are considered ungrammatical and the error detection

system allows fewer n-grams to pass as grammatically correct, leading to more cases being flagged as errors and to a higher recall score. At the same time, more language structures, that are rarer but actually grammatical, are flagged as errors resulting in lower precision.

In setting 2 for both test sets, there is no difference in precision, recall or f-score between cutoffs 0 and 1, which means that they both flag the same sentences as erroneous. This does not happen in setting 1 because having to locate the error correctly within a sentence creates enough of a difference between cutoffs 0 and 1. Additionally, the f-score raises only slightly when moving towards higher cutoff values in setting 1. In setting 2, the f-score raises more dramatically because of the aggressive ascend in the recall. When the cutoff value is higher, more n-grams in general are considered ungrammatical leading to more language structures being flagged as errors. The error can be anywhere in the sentence for the error flagging to be considered correct and more sentences are flagged correctly in setting 2. However, the error location within a sentence has to be correct in setting 1, and raising the cutoff values makes it more likely for the checker to flag an error early in the sentence before an actual error. This is not enough for the error detection to be considered correct, and thus, the rise in the recall is milder in setting 1.

6.1.5 POS-tags

The performance of each of the setups using different POS-tags in their n-gram sets are compared to each other. There are five POS-tags each of which consisting of different variations of three pieces of linguistic information: word class, grammatical number, and grammatical case. Word-class-tag (wc) contains only word class, whole-tag (wt) has all three components, number/case-tag (nc) consists of number and case, number-tag (nu) has only number and case-tag (ca) has only case.

Table 17 shows the average scores for setups using each of the POS-tags. In both settings and both test sets, number-tag achieves the best precision and whole-tag has the highest recall and f-score except in setting 1 using normal test set, where number/case-tag has the highest f-score.

Out of all of the POS-tags, number-tag has the least amount of options it can appear as because it can consists of only a single selection from eight different pieces of grammatical number information as shown previously in table 1 in section 3.2. This causes number-tag to be the least restrictive tag, and it flags the least errors in general, which is reflected in the lowest recall scores. When a system using number-tags actually does flag an error, it is

also quite sure of its decision, at least more sure than systems using the other tags.

	Setting 1	wc	wt	nc	nu	ca
Normal	precision	0.253	0.204	0.223	**0.268**	0.258
test set	recall	0.288	**0.386**	0.351	0.265	0.294
	f-score	0.264	0.267	**0.270**	0.263	0.269

	Setting 2	wc	wt	nc	nu	ca
Normal	precision	0.562	0.505	0.517	**0.569**	0.539
test set	recall	0.639	**0.957**	0.814	0.565	0.614
	f-score	0.585	**0.660**	0.626	0.561	0.562

	Setting 1	wc	wt	nc	nu	ca
NUN	precision	0.339	0.215	0.246	**0.391**	0.345
test set	recall	0.244	**0.399**	0.339	0.202	0.246
	f-score	0.259	**0.279**	0.277	0.252	0.261

	Setting 2	wc	wt	nc	nu	ca
NUN	precision	0.682	0.509	0.537	**0.731**	0.632
test set	recall	0.504	**0.946**	0.742	0.390	0.463
	f-score	0.527	**0.660**	0.605	0.480	0.486

Table 17: The average scores of test setups using different POS-tags

Conversely, whole-tag has the most linguistic representations it can appear as. Whole-tag can consist of one of 13 word classes, one of 15 grammatical cases and one of 8 grammatical numbers. Taking into account that case and number can also be empty, the total number of possible whole-tags is in theory $13 * 16 * 9 = 1872$. The actual number is not this high as some combinations of word class, number and case cannot occur, for example, nominals do not have grammatical person. In any case, systems using whole-tags are the most restrictive, as the whole-tags in the n-grams of a given input sentence have to match whole-tags in the reference n-gram sets. This causes whole-tag setups to flag the most errors in general, which in turn raises the recall score.

In fact, the order of the highest precision scores follows the order of how restrictive the POS-tags are based on the number of forms they can represent. Number-tag has 9 different forms, word-class-tag has 13, case-tag has

16, number/case-tag has 144 and whole-tag has 1872. The order of highest precision is number-tag, word-class-tag, case-tag, number/case-tag, and whole-tag, although case-tag has slightly higher precision than word-class-tag in setting 1. The order of highest recall is the order of the highest precision in reverse: whole-tag, number/case-tag, case-tag, word-class-tag, and number-tag, although word-class-tag has higher recall than case-tag in setting 2.

6.1.6 Best setups

For each setting and each test set, the best grammar detection system setups are reported. Each setup uses FTB1 or FTB3, WB or WoB, CMP or CAP, one of the five cutoff values, and one of the five POS-tags. The setups that performed the best with respect to precision, recall, and f-score are presented. Based on the observations made in previous sections, the setups with the highest precision should use FTB3, WoB, CAP, cutoff 0 and number-tags, and the setups with highest recall and f-score should use FTB1, WoB in setting 1 but WB in setting 2, CMP, cutoff 30 and whole-tags.

Setup	precision	recall	f-score
FTB3 WoB CAP cutoff-30 nu	**0.343**	0.274	0.304
FTB1 WoB CMP cutoff-10 ca	0.267	**0.468**	0.340
FTB1 WoB CMP cutoff-10 ca	0.267	0.468	**0.340**

Table 18: Best setups for normal test set in setting 1

Table 18 displays the performance of the best setups for normal test set in test setting 1. The results are somewhat higher than the overall average results, as shown previously in section 6.1 table 12, but they are still quite poor. Highest precision, 0.343, is achieved by a setup that uses FTB3, WoB, CAP, and number-tag, which was expected. However, the setup uses cutoff 30, which is the opposite of what the average results suggest. For normal test set, the differences in average precision values between different cutoff points are quite minor, and this setup seems to be just an outlier. The setup that reaches the highest recall, 0.468, and simultaneously the highest f-score, 0.340, uses FTB1, WoB, CMP, cutoff 10 and case-tag. Cutoff 30 has slightly higher recall than cutoff 10 in the average results. Case-tag performs the third best in recall and the second best in f-score in the average results. Otherwise, this setup uses the expected components. It is worth noting, that the setups with the highest precision and recall have very poor recall and

precision values respectively, and these setups would be practically unusable in a real-world setting.

Setup	precision	recall	f-score
FTB3 WoB CAP cutoff-30 wc	**0.598**	0.475	0.530
FTB1 WB CAP cutoff-10 wt	0.501	**1.000**	0.668
FTB1 WB CAP cutoff-0 wt	0.505	0.990	**0.668**

Table 19: Best setups for normal test set in setting 2

Table 19 shows the best setups for normal test set in setting 2, where whole sentences are tagged as erroneous. The setup with the highest precision value, 0.598, uses FTB3, WoB, CAP, cutoff 30 and word-class-tag, which is again almost exactly what was to be expected. Different cutoff values were previously explained to have only minor differences in normal test set and word-class-tag has the second highest precision value in normal test set in setting 2. This setup has a recall of 0.475 and an f-score of 0.530, so the checker finds little less than half of the errors and correctly flags them a little over half the time. A hypothetical grammar checker, that flags every second sentence as an error, would reach precision of 0.500, recall of 0.500 and f-score of 0.500 in this test set where half of the samples contain errors and half do not. This setup performs only slightly better than the hypothetical grammar checker and does not seem to have any actual grammar error detection capabilities.

The setup that has the highest recall, 1.000, uses FTB1, WB, CAP, cutoff 10 and whole-tag, which was otherwise expected except for CAP. There are multiple other setups that reached a recall of 1.000 with cutoffs 10 and 30, all of which use whole-tag. However, this setup does not flag every single sentence as an error as it reached a precision value of over half: 0.501. It also achieves the highest f-score with 0.668, but the setup, that uses FTB1, WB, CAP, cutoff 10 and whole-tag, is chosen to represent the highest f-score, as it has slightly different, although similar, precision and recall values.

Setup	precision	recall	f-score
FTB3 WoB CAP cutoff-0 nu	**0.647**	0.202	0.308
FTB1 WoB CMP cutoff-0 nc	0.266	**0.488**	0.344
FTB1 WoB CMP cutoff-10 ca	0.280	0.467	**0.350**

Table 20: Best setups for NUN test set in setting 1

Table 20 shows the best setups for NUN test set in setting 1. The best setups achieve higher precision but similar recall and f-score as in the normal test set in setting 1. In NUN test set, there are fewer words that Omorfi does not recognize, which helps the checker to avoid flagging unknown errors in places where they should not be flagged, which in turn raises the precision. The setup with the highest precision, 0.647, uses the components FTB3, WoB, CAP, cutoff 0 and number-tag, which are exactly the ones that have the highest average precision scores in the previous sections. The setup with the highest recall, 0.488, uses FTB1, WoB, CMP, cutoff 0, and number/case-tag. Number/case-tag reaches the second highest recall in the average measurements but cutoff 0 performs the worst in this regard, so it is surprising to see it in the setup with the highest recall. The best f-score setup uses FTB1, WoB, CMP, cutoff 10, and case-tag. Based on the average score results, FTB1, WoB, and CMP were expected, and cutoff 10 and case-tag are not too surprising, as they are the second and third best components respectively when reaching for high f-score.

Setup	precision	recall	f-score
FTB3 WoB CAP cutoff-0 nu	**0.857**	0.267	0.408
FTB1 WB CMP cutoff-10 wt	0.500	**1.000**	0.667
FTB1 WB CAP cutoff-0 wt	0.506	0.990	**0.669**

Table 21: Best setups for NUN test set in setting 2

The best setups in NUN test set in setting 2 reach the highest precision, recall and f-score out of all other setups, as shown in table 21. This was expected because this setting has the most relaxation with unknown foreign proper names removed and the checker having to only label whole sentences instead of exactly locating the grammar errors. The setup with the highest precision uses FTB3, WoB, CAP, cutoff 0 and number-tag, which correlates with the results from the average score measurements. The precision value, 0.857, seems moderately impressive at first glance, but further examination shows that the same setup has a recall value of 0.267 and an f-score of 0.408. In other words, the setup flags sentences as erroneous very rarely, but when it does, it is quite sure that the sentence actually is erroneous. The setup with the best recall, 1.000, uses FTB1 WB, CMP, cutoff 10 and whole-tag, which, again, are the expected components, except for the cutoff value, which should be 30 according to the average results. The setup with the highest f-score, 0.669, uses FTB1, WB, CAP, cutoff 0 and whole-tag, which is somewhat surprising as it uses CAP instead of CMP and cutoff 0 instead of cutoff 30.

Both, the high recall setup and the high f-score setup, tag practically every single sentence as an error, so they reach perfect recall and precision of about 0.500. Both also have significantly higher f-score that the setup with precision 0.857, but one could argue that the high precision setup is more user-friendly than the other two setups. Firstly, flagging everything as an error gives no useful information to the user of the grammar checker. Secondly, favoring low recall but high precision makes the checker less annoying in real word usage, as it is less likely to interrupt the writing experience with falsely flagged errors. Although, in this case, the recall of 0.267 is too low for the setup to be considered actually useful.

All in all, the best setups have quite poor performance. Using the relaxed NUN test set with unknown names removed helps in raising precision. In setting 2, where the checker flags whole sentences instead of exact locations in a sentence, the results are overall better. Some setups reached a recall of 1.000, but at the same time, their precision is 0.500, which means that they flag every sentence in the test set as erroneous, and they do not show any actual grammar error detection capabilities. These kinds of setups are also the ones with the highest f-scores. The most interesting setup is the one that reached a precision of 0.857 in NUN test set in setting 2. Even though it has a very low recall value, 0.267, it has some resemblance of a grammar checker that has been tuned to have a low recall but high precision value. Manual observation of the results shows that there are no setups that reach a better harmony of precision and recall than the ones that flag every sentence, which is also suggested by the fact that these setups have the highest f-scores. A setup, that comes somewhat close, uses FTB1, WB, CMP, cutoff 1 and word-class-tag in NUN test set in setting 1. It has a precision of 0.576, recall of 0.679 and f-score of 0.623, so it does not flag every single sentence as an error, it finds a good majority of the errors and the precision is slightly higher than the precision of the setups with a recall of 1.000.

There seems to not be any performance reports for Finnish grammar checkers, that the error detection system of this thesis could be compared to. Nazar and Renau (2012) have a similar approach to grammar checking for Spanish, but they use Google Books N-grams as their reference corpus and plain text n-grams instead of POS n-grams. They report their checker to achieve precision of 0.618, recall of 0.547 and f-score of 0.581 in error detection. Some of the setups in test setting 1 in my work have similar results, although Nazar and Renau themselves admit that their results are quite poor. The test setups in setting 2 cannot be compared to these results, because they flag whole sentences while the reported values are measured on localized errors. Sjöbergh (2009) uses the Swedish internet as the reference n-gram corpus for

checking grammar errors in Swedish text. He reports a precision value of 0.913 on detecting errors in second language learner essays. Recall value is not reported, but based on the amount of errors detected by other checkers displayed in the paper, the recall value can be 0.689 at maximum, in which case, f-score would be 0.785. None of the grammar checker setups in the system presented in my work were able to achieve nearly as good results, even in a relax test setting where whole sentences are flagged.

6.2 Error types

The average error type detection performances are reported for each setup using one of the five POS-tags. Additionally, the grammar checker setups that were best in finding each error type are displayed. The performances are measured as the proportion of errors with a given error type found by the error detection system. The different error types were described previously in section 5.1.1. It is important to note that the grammar checking system does not label the errors it finds with an error type. Instead, the error type detection here simply means that the system correctly flagged an error, and the error has been labeled with a given error type in the test data. For example, if a checker setup is reported to have 50% performance in finding "wrong case" errors, the setup correctly flagged 50% of the errors, which have the label "wrong case" in the test data, while leaving 50% unflagged.

The error type performance results are reported only for each POS-tag but not for other grammar checker components for two reasons. First, the different POS-tags containing varying linguistic information were specifically constructed with the hopes that they would capture different types of errors. For example, number tags should have a good picture of how grammatical number behaves and should be able to find more number related errors than case tags, which, in turn, should be better in finding grammatical case errors. Secondly, there are no interesting differences between the other grammar checker components. The only occurring phenomenon is that the checkers that have a high recall, i.e. the checkers that use FTB1, WB, CMP, and high cutoff threshold, perform better than checkers with high precision in detecting different error types. The checker setups with high recall naturally flag more language structures as errors and they are bound to find more errors in general. Previously, the average results showed that the setups, that use whole-tags have the highest recall values, which suggests that whole-tags should perform the best in detecting different error types in general, unless having a specific set of linguistic information components in some other tag

overrides this.

6.2.1 POS-tags

The average error type detection proportions for all setups using one of the five POS-tags are shown in table 22 for normal test set and in table 23 for NUN test set. Each of the five POS-tags contain different variations of linguistic information. Word-class-tag (wc) only has information on word class; whole-tag (wt) has word class, number and case; number/case-tag (nc) has number and case; number tag (nu) has number and case tag (ca) has case. The intuition is that POS-tags, that carry a given type of linguistic information, should be able to detect that type of errors. The clearest cases in this regard are number-tags which should detect "wrong number" errors, and case-tags which should detect "wrong case" errors.

	wc	wt	nc	nu	ca
extra structure	10.1%	**21.4%**	13.0%	5.0%	6.8%
foreign word	46.1%	35.0%	39.1%	**48.8%**	47.6%
missing predicate	34.0%	**60.3%**	48.1%	23.7%	28.9%
missing structure	16.4%	**31.7%**	27.3%	12.7%	16.2%
unknown word	51.4%	35.9%	44.1%	53.9%	**55.6%**
wrong case	11.8%	**25.5%**	19.2%	10.9%	13.4%
wrong case and number	4.4%	**19.7%**	13.4%	2.2%	10.6%
wrong number	3.8%	8.9%	**9.2%**	4.2%	1.2%
wrong person	8.4%	15.2%	**16.8%**	10.5%	9.8%
wrong structure	40.1%	**49.5%**	48.0%	37.9%	41.5%
wrong word class	9.7%	**18.4%**	16.6%	11.2%	18.1%
wrong word order	10.0%	**25.8%**	16.7%	0.0%	11.7%

Table 22: The average error type detection proportions for each POS-tag in normal test set

The first observation to be seen from table 22 is that whole-tag has the best detection performances for most of the error types. However, whole-tag performs the worst in finding foreign word and unknown word errors, whereas number-tag and case-tag, both of which are associated with high precision and low recall values, find the largest proportion of errors with these error types. Foreign word and unknown word errors are the easiest to detect because, in both of these cases, Omorfi tags a word with an "unknown" label during error detection. POS n-grams that include "unknown" tags are

always flagged as errors since "unknown" tags do not appear in the reference set n-grams. If a foreign word or an unknown word is further down a given sentence, whole-tag is likely to falsely flag an error before reaching the actual error. Number-tags and case-tags, on the other hand, have low recall and are less likely to flag errors before the actual foreign word or unknown word error, which are then reached and correctly flagged. These error types being the easiest to detect is also reflected by them being among the most detected categories (foreign word: 48.8%, and unknown word: 55.6%).

Number-tag and case-tag perform poorly in detecting grammatical case, number and person related errors when compared to whole-tag and number/case-tag. Number-tag and case-tag have low recall while whole-tag and number/case-tag have a high recall in the average measurements, so it seems that just flagging more errors is more advantageous than having the specific linguistic information to detect a certain error type. However, case-tag is better than number-tag in finding grammatical case errors (13.4% vs 10.9%) and number-tag is better than case-tag in finding number errors (4.2% vs 1.2%) and person errors (10.5% vs 9.8%). These values are quite low, but they give some indication that proper linguistic information could matter in finding correct error types. Comparing number-tag and case-tag against each other in this regard is meaningful as they have been previously shown to have similar precision and recall values while comparing them to whole-tag is slightly dubious because it has higher recall and lower precision.

	wc	wt	nc	nu	ca
extra structure	9.2%	**22.6%**	14.8%	6.5%	9.9%
foreign word	46.8%	33.8%	39.9%	**50.3%**	49.4%
missing predicate	26.6%	**61.1%**	46.3%	14.6%	21.8%
missing structure	13.2%	**33.0%**	26.1%	7.6%	13.5%
unknown word	49.6%	36.9%	43.9%	51.3%	**54.2%**
wrong case	8.9%	**26.8%**	18.9%	6.5%	9.9%
wrong case and number	4.1%	**20.3%**	15.0%	3.1%	11.9%
wrong number	4.7%	**11.9%**	9.4%	3.6%	1.2%
wrong person	7.9%	15.9%	**18.4%**	13.0%	10.5%
wrong structure	32.7%	**51.6%**	45.1%	26.3%	32.1%
wrong word class	9.7%	**18.4%**	16.2%	11.6%	18.1%
wrong word order	10.0%	**25.0%**	15.8%	0.0%	11.7%

Table 23: The average error type detection proportions for each POS-tag in NUN test set

In NUN test set, where unknown foreign names are removed, the results are very similar to the results in normal test set as seen in table 23. The proportions of found errors are slightly higher for most of the error types, but the relationships between each POS-tag stay the same. As shown previously, using different POS-tags in normal test set and in NUN test set yield similar precision and recall values, but with NUN test set having somewhat higher precision. This explains why error type detection is slightly, but only slightly, better in NUN test set than in normal test set for most of the error types. Removing unknown foreign names from the test set does not dramatically improve detecting foreign word or unknown word errors as one's first intuition might be. Instead, the purpose of the NUN test set is to prevent the checker from flagging errors where they should not be flagged and allowing the error detection algorithm to move on in the sentence to find other errors.

6.2.2 Best setups

In the following, the best grammar checker setups in finding each of the error types are reported. Based on previous observations, most of these setups should utilize components that achieve high recall values, i.e. FTB1, WoB, CMP, high cutoff values and whole-tag.

	Value	Setup
extra structure	38.2%	FTB1 WoB CAP cutoff-30 wc
foreign word	85.4%	FTB3 WoB CMP cutoff-30 nu
missing predicate	72.2%	FTB1 WB CMP cutoff-30 wc
missing structure	45.2%	FTB1 WoB CMP cutoff-5 nc
unknown word	96.5%	FTB3 WoB CMP cutoff-5 wc
wrong case	33.5%	FTB1 WoB CMP cutoff-0 wt
wrong case and number	50.0%	FTB1 WoB CMP cutoff-5 nc
wrong number	43.8%	FTB1 WoB CMP cutoff-0 nc
wrong person	35.7%	FTB1 WoB CMP cutoff-0 nc
wrong structure	63.3%	FTB1 WB CMP cutoff-0 nc
wrong word class	37.5%	FTB3 WoB CMP cutoff-0 wt
wrong word order	66.7%	FTB1 WoB CMP cutoff-0 ca

Table 24: The best error type detection setups in normal test set

Table 24 shows the best setups in detecting each error type in normal test set. Most of the best setups do use FTB1, WoB, and CMP as predicted,

but cutoff 30 is used only for finding foreign words, missing predicates and missing structures and the rest of the best setups use either cutoff threshold 0 or 5. Whole-tag and number/case-tag, which are associated with high recall values, are used in finding seven of the twelve error types. The best setup in finding wrong case errors (34.6%) uses wt-tags, and the best setups in finding wrong number (43.8%) and wrong person (42.9%) use number/case-tag. This follows the trend shown in the average results, where tags, that achieve high recall, also find the most errors of these types.

Wrong case is the most difficult error type to find with 34.6%. This could be due to the fact that Finnish has intricate morphology and extensive generative properties as explained by Karlsson (2008). In some cases, a certain sequence of case ending is perfectly grammatical, while sometimes, the same sequence can be incorrect depending on the choice of words, for example in idiomatic expressions. Other difficult to find error types are wrong person with 35.7% and wrong word class with 37.5%. Most of the wrong person errors in the test data are cases, where a verb is in a certain form of singular while it should be in the corresponding plural, or vice versa, based on the subject of the sentence. Sometimes the subject is a list of actors, in which case it is easy to see why the error would not be detected. For example, the sentence "John, Tim and Matt runs" is ungrammatical, but if the checker only sees the last two words, "Matt runs", the error would not be flagged. The grammar checker system does utilize n-grams up to length five, but even then the ability to detect this kind of errors depends on the existence of appropriate examples in the training data. The difficulty of finding wrong word class errors can be seen for example in cases, where the word-class is noun while it should be adjective, which is hard for the grammar checker to detect because they behave similarly linguistically.

The easiest error types for the best grammar checker setups to detect are unknown word with 96.5% and foreign word with 85.4%. This was expected and the reason was described in the previous section 6.2.1. The next easiest to find error is missing predicate with 72.2%. The best setup in this regard uses word-class-tag, which is not associated with particularly high recall and it only contains information on the word class. This indicates that the system could in some degree be able to detect the absence of a verb in a sentence and flag it as an error. The good performance is also aided by missing predicate errors spanning the section of the sentence where the predicate could occur, which is usually multiple words. The error is considered to be correctly detected if the error location received from the grammar checker overlaps with the actual location of the error even partially. The next error types, where the best setups fare well, are wrong word order with 66.7%

and wrong structure with 63.3%, both of which produce obscure POS n-gram combinations which have a high likelihood of not being found in the reference set of n-grams. However, there are only three sentences in the test data, where the error type is wrong word order, and the checker could have accidentally found the two of the three errors.

The results of the best setups in detecting each error type in NUN test set are shown in table 25. There are no major differences between the results in normal test set and NUN test set. The performances are slightly better for NUN test set and the best setups for each error type use almost exactly the same components as the best setups in normal test set. The better performance in NUN test set can, again, be explained by the grammar detection system being able to avoid falsely detecting unknown word errors in the beginning portions of the test sentences, and being able to move on to reach the actual errors.

	Value	Setup
extra structure	35.3%	FTB1 WoB CAP cutoff-30 wc
foreign word	87.5%	FTB3 WoB CMP cutoff-30 nu
missing predicate	74.4%	FTB1 WB CMP cutoff-30 wc
missing structure	45.2%	FTB1 WoB CMP cutoff-30 nc
unknown word	96.5%	FTB3 WoB CMP cutoff-5 wc
wrong case	34.6%	FTB1 WoB CMP cutoff-0 wt
wrong case and number	50.0%	FTB1 WB CMP cutoff-0 ca
wrong number	43.8%	FTB1 WoB CMP cutoff-0 nc
wrong person	42.9%	FTB1 WoB CMP cutoff-0 nc
wrong structure	64.9%	FTB1 WB CMP cutoff-0 nc
wrong word class	37.5%	FTB3 WoB CMP cutoff-0 wt
wrong word order	66.7%	FTB1 WoB CMP cutoff-0 ca

Table 25: The best error type detection setups in NUN test set

All in all, the checker performs well only in detecting errors caused by foreign words and unknown words. This, however, is not a terribly impressive feat as similar results could be easily achieved using only Omorfi, or a robust dictionary, to see whether a token is an actual Finnish word or not. Detecting missing predicates works relatively well comparing to other error types, which gives some indication of the system being able to detect grammar errors. Using specific linguistic information in POS-tags to detect specific types of errors did not have the intended effect. Case-tag is fairly poor in detecting wrong case errors, and number-tag is similarly bad in finding wrong num-

ber and wrong person errors, although case-tag is better than number-tag in detecting wrong case errors and vice versa. Further, it is important to remember that the error type detected by the system is to some degree dictated by the error's position in a sentence. Especially for very strict setups, if the error is located towards the end of a sentence, it is likely that the checker falsely flags some other error before reaching the actual error. Another factor is that some errors span long word sequences, while others are located in a single word. For example, missing predicate errors cover the section of the phrase where the predicate is expected to be, and the checker has a good chance of flagging some part of the error accidentally. Conversely, wrong case and wrong number errors are located in single words making them difficult to detect for the grammar checker by chance.

7 Conclusions

This work evaluates the possibility of using a POS n-gram based approach in detecting grammar errors in Finnish text. Precision, recall and f-score values, as well as the types of errors that the presented method is able to detect, are examined. In this method, POS n-grams are gathered from an annotated corpus to a set, which is used to check grammar by seeing whether an input sentence's POS n-grams exist in that set. This method is chosen for two main reasons: first, the method is simple in principle and easy to implement, and second, it is able to capture a larger variety of errors than a rule-based checker without having to manually write extensive amounts of grammar rules.

The main finding is that the proposed implementation of a POS n-gram based grammar error detection system does not perform very well in detecting errors in Finnish text. In a test setting, where the checker has to find the location of an error within a sentence, the tested grammar checker setups have low precision and recall values. In a relaxed test setting, where the system flags whole sentences as erroneous, some grammar checker setups are able to achieve either high precision or high recall. Each setup's performance is largely dictated on how restrictive the setup is. If the checker is very strict and flags every single test sentence as an erroneous one, a recall value of 1.000 is easily achieved, but precision will stay low. On the other hand, if the checker flags very few errors in general, the recall is kept down, but it is able to reach a higher precision value. Additionally, setups using different POS-tags are not very effective in finding specific error types. Whole-tags, which are the most strict tags as they contain the most linguistic information,

perform the best in finding almost all of the error types. Further, the error type detection performance values are low for most of the categories. The presented approach of using POS n-grams does not work very well for Finnish text, which can be explained by the rich morphology and extensive generative features.

In addition to the negative main results, there are two positive findings to be noted. First, the grammar checker setup, that uses FTB3, WoB, CAP, cutoff 0 and number-tag, is able to achieve a precision of 0.857, although in a test setting, where foreign proper names are removed and the checker only flags whole sentences. Even though the recall value of the setup is only 0.267, reaching this high precision requires the system to have some kind of knowledge of what a grammar error is, unlike a system that reaches perfect recall by flagging everything as an error. Second, the fact that case-tags are better at finding case errors than number-tags, and that number-tags are better at finding number errors than case-tags, indicates that having certain linguistic information in the POS-tags can help in finding specific types of errors. Whole-tags and number/case-tags have better success in finding both number and case errors, but they flag more errors in general which helps them in finding all error types. Number-tags and case-tags have similar recall scores compared to each other, although much lower than what whole-tags or number/case-tags have, which justifies the comparison between number-tags and case-tags.

There are ways to improve the error detection system presented in this work. A feature, that would be beneficial to have in the pipeline, is named entity recognition (NER). This would help in dealing with foreign proper names that are abundant in the test data and that are not recognized by Omorfi. Although test setting 2 simulates including NER by having foreign proper names replaced by Finnish ones, an actual NER system is naturally required in a real-world grammar checking scenario in order to achieve the same effect. Work on NER systems, that could potentially work for Finnish, has been done among others by Agerri and Rigau (2016), who introduce a robust multilingual NER system, and by Kettunen et al. (2016), who search for named entities in Finnish historical newspapers. Including a disambiguation component to the grammar error detection system would also help with the grammar checking quality. Currently, the system has two very extreme ways of handling ambiguous sentence structures: one, where only one interpretation based on simple statistics is checked, and another, where all combinations of all possible POS-tags of a sentence are checked. Instead of using one of these approaches, a disambiguator could list a certain number of the most probable interpretations, each of which would then be checked. This kind of

grammar checker would not be nearly as strict as one that checks only a single interpretation based on simple statistics. However, it would be much more restrictive than a system that checks all possible POS-tag combinations, a major portion of which are potentially ungrammatical or at least very rare. Early work on disambiguation for Finnish has been done by Koskenniemi (1990) and more recently by Robertson (2019). Omorfi also includes a disambiguation feature (Pirinen, 2015). Another obvious way to improve the system is to use a larger morphologically annotated training corpus. One option would be the Finnish Internet parsebank corpus by Kanerva et al. (2014) containing 1.5 billion tokens in 116 million sentences, which dwarfs FTB3's 76 million tokens in 4 million sentences. Dealing with data of this size naturally brings computational performance issues, and the reference n-gram lookup would have to be conducted in a more clever way than just checking whether an n-gram is included in the whole set of correct n-grams. Instead of having a huge corpus to cover everything, another approach would be to train a checker for a specific text domain with a certain type of training data. However, this approach would require multiple different morphologically annotated corpora, which are most likely not easily available. This issue can be circumvented by including a POS-tagger in the n-gram extraction phase.

The state of the art approach to grammar checking is to use neural networks. In most cases, the latest systems do not only detect grammar errors but correct the errors as well. The grammar error correction (GEC) task can be treated as a machine translation task, where ungrammatical sentences are the source language and correct sentences are the target language. An example of such work has been done by Chollampatt and Ng (2018), who apply a convolutional encoder-decoder architecture to grammar error correction. However, neural network based GEC requires a large parallel corpus of ungrammatical sentences and their correct counterparts for training. The issue can be mitigated by producing noisy data from a clean monolingual corpus, as described by Xie et al. (2018), who introduce beam search based data noising methods. A GEC system for Finnish would most likely outperform the quite simple error detection approach used in the system presented in this work, while simultaneously producing corrections of the found errors.

The work in this thesis introduces a simple and easy to implement statistical method for grammar error detection for Finnish. Although there are some signs of error detection capabilities in relaxed test settings, the system achieves quite poor results overall. The proposed approach could be improved and made more robust, but a more effective way of trying to reach better performance would be to adopt the state of the art methods of using neural networks.

References

Rodrigo Agerri and German Rigau. Robust multilingual named entity recognition with shallow semi-supervised features. *Artificial Intelligence*, 238: 63–82, 2016.

Jahangir Md. Alam, Naushad UzZaman, and Mumit Khan. *N-gram based Statistical Grammar Checker for Bangla and English.* Center for Research On Bangla Language Processing, 2007.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally Normalized Transition-Based Neural Networks. *CoRR*, abs/1603.06042, 2016. URL `http://arxiv.org/abs/1603.06042`.

Eric Steven Atwell. How to Detect Grammatical Errors in a Text Without Parsing It. In *Proceedings of the Third Conference on European Chapter of the Association for Computational Linguistics*, EACL '87, pages 38–45, Stroudsburg, PA, USA, 1987. Association for Computational Linguistics. doi: 10.3115/976858.976865. URL `https://doi.org/10.3115/976858.976865`.

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Johnny Bigert and Ola Knutsson. Robust Error Detection: A Hybrid Approach Combining Unsupervised Error Detection and Linguistic Knowledge. In *Proceedings of Romand 2002, Robust Methods in Analysis of Natural language Data*, pages 10–19, 2002.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based N-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479, December 1992. ISSN 0891-2017. URL `http://dl.acm.org/citation.cfm?id=176313.176316`.

Michelle Cavaleri and Saib Dianati. You want me to check your grammar again? The usefulness of an online grammar checker as perceived by students. *Journal of Academic Language and Learning*, 10(1), 2016. ISSN 1835-5196. URL `http://www.journal.aall.org.au/index.php/jall/article/view/393/246`.

David Chiang. A Hierarchical Phrase-based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting on Association*

for Computational Linguistics, ACL '05, pages 263–270, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219873. URL https://doi.org/10.3115/1219840.1219873.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR*, abs/1409.1259, 2014. URL http://arxiv.org/abs/1409.1259.

Martin Chodorow and Claudia Leacock. An Unsupervised Method for Detecting Grammatical Errors. In *1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, 2000.

Shamil Chollampatt and Hwee Tou Ng. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Michael Collins, Brian Roark, and Murat Saraclar. Discriminative Syntactic Language Modeling for Speech Recognition. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 507–514, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219903. URL https://doi.org/10.3115/1219840.1219903.

Daiga Deksne and Raivis Skadiņš. CFG based grammar checker for Latvian. In *NODALIDA 2011 Conference Proceedings*, pages 275–278, 2011.

Rickard Domeij, Ola Knutsson, Johan Carlberger, and Viggo Kann. "granska–an efficient hybrid system for swedish grammar checking". In *Proceedings of the 12th Nordic Conference of Computational Linguistics (NODALIDA 1999)*, pages 49–56, Trondheim, Norway, December 2000. Department of Linguistics, Norwegian University of Science and Technology, Norway. URL https://www.aclweb.org/anthology/W99-1005.

Dan Garrette and Jason Baldridge. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 138–147, 2013.

A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013. doi: 10.1109/ICASSP.2013.6638947.

Alex Graves and Navdeep Jaitly. Towards End-to-End Speech Recognition with Recurrent Neural Networks. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Kristin Hagen, Janne Bondi Johannessen, and Pia Lane. Some problems related to the development of a grammar checker. In *Proceedings of the 13th Nordic Conference of Computational Linguistics (NODALIDA 2001)*, 2001.

Auli Hakulinen, Maria Vilkuna, Riitta Korhonen, Vesa Koivisto, Tarja Riitta Heinonen, and Irja Alho. *Ison suomen kieliopin verkkoversio (VISK)*. Suomalaisen Kirjallisuuden Seura, Helsinki, 2004. URL `http://scripta.kotus.fi/visk/`. Accessed: 2019-25-06.

Peter A Heeman. POS tags and decision trees for language modeling. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

G. E. Heidorn, K. Jensen, L. A. Miller, R. J. Byrd, and M. S. Chodorow. The EPISTLE Text-critiquing System. *IBM Systems Journal*, 21(3):305–326, September 1982. ISSN 0018-8670. doi: 10.1147/sj.213.0305. URL `http://dx.doi.org/10.1147/sj.213.0305`.

Elli Heikkilä and Selene Peltonen. Immigrants and integration in Finland. *Survey: About the Situation of Immigrants and Refugees in Six Baltic Sea States. Developed within the framework of the European Community Action, SOCRATES*, 2002.

Nal Kalchbrenner and Phil Blunsom. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013.

Jenna Kanerva, Juhani Luotolahti, Veronika Laippala, and Filip Ginter. Syntactic N-gram Collection from a Large-Scale Corpus of Internet Finnish. In *Proceedings of the Sixth International Conference Baltic HLT*, 2014.

Fred Karlsson. *Finnish: An Essential Grammar* . Routledge, London, 2008.

Slava M. Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-53:400–401, 3 1987.

Kimmo Kettunen, Eetu Mäkelä, Teemu Ruokolainen, Juha Kuokkala, and Laura Löfberg. Old Content and Modern Tools - Searching Named Entities

in a Finnish OCRed Historical Newspaper Collection 1771-1910. *CoRR*, abs/1611.02839, 2016. URL http://arxiv.org/abs/1611.02839.

Philipp Koehn. Statistical Machine Translation, 2005. URL www.statmt.org. Accessed: 2018-30-3.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical Phrase-based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073462. URL https://doi.org/10.3115/1073445.1073462.

Kimmo Koskenniemi. Finite-state Parsing and Disambiguation. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90, pages 229–232, Stroudsburg, PA, USA, 1990. Association for Computational Linguistics. doi: 10.3115/997939.997979. URL https://doi.org/10.3115/997939.997979.

Roland Kuhn and Renato De Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 570–583, 7 1990.

Nay Yee Lin, Khin Mar Soe, and Ni Lar Thein. Developing a Chunk-based Grammar Checker for Translated English Sentences. In *Proceedings of the 25th Pacific Asia Conference on Language, Information and Computation*, pages 245–254, 2011.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. NAACL, May 2015. URL https://www.microsoft.com/en-us/research/publication/representation-learning-using-multi-task-deep-neural-networks-for-semantic-classification-and-information-retrieval/.

Zhuo-Ran Liu and Yang Liu. "exploiting unlabeled data for neural grammatical error detection". *Journal of Computer Science and Technology*, 32 (4):758–767, Jul 2017. ISSN 1860-4749. doi: 10.1007/s11390-017-1757-4. URL https://doi.org/10.1007/s11390-017-1757-4.

Nina H Macdonald. Human factors and behavioral science: The UNIX™ Writer's Workbench software: Rationale and design. *Bell System Technical Journal*, 62(6):1891–1908, 1983.

Christopher D Manning. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International conference on intelligent text processing and computational linguistics*, pages 171–189. Springer, 2011.

James H Martin and Daniel Jurafsky. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/Prentice Hall Upper Saddle River, 2009.

Bernard Merialdo. Tagging English Text with a Probabilistic Model. *Computational Linguistics*, 20(2):155–171, June 1994. ISSN 0891-2017. URL `http://dl.acm.org/citation.cfm?id=972525.972526`.

Anneli Miettinen and Tiina Helamaa. Maahanmuuttajien määrä, 2019. URL `http://www.vaestoliitto.fi/tieto_ja_tutkimus/vaestontutkimuslaitos/tilastoja/maahanmuuttajat/maahanmuuttajien-maara/`. Accessed: 2019-18-04.

David R. H. Miller, Tim Leek, and Richard M. Schwartz. A Hidden Markov Model Information Retrieval System. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 214–221, 1999.

Teruko Mitamura and Eric Nyberg. Controlled English for Knowledge-Based MT: Experience with the KANT System. In *Proceedings of TMI-95*, 11 1995.

Rogelio Nazar and Irene Renau. Google Books N-gram Corpus used as a grammar checker. In *Proceedings of the EACL 2012 Workshop on Computational Linguistics and Writing*, pages 27–34, 2012.

Jan Niehues and Muntsin Kolss. A POS-based Model for Long-range Reorderings in SMT. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, pages 206–214, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1626431.1626472`.

T. R. Niesler and P. C. Woodland. A variable-length category-based n-gram language model. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 164–167 vol. 1, May 1996. doi: 10.1109/ICASSP.1996.540316.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Tommi A Pirinen. Omorfi—free and open source morphological lexical database for finnish. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 313–315, 2015.

F. Pla, A. Molina, E. Sanchis, E. Segarra, and F. García. "language understanding using two-level stochastic models with pos and semantic units". In Václav Matoušek, Pavel Mautner, Roman Mouček, and Karel Taušer, editors, *Text, Speech and Dialogue*, pages 403–409, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44805-1.

Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st ACM Conference on Research and Development in Information Retrieval (SIGIR'98)*, 1998.

Stephen D. Richardson and Lisa C. Braden-Harder. The Experience of Developing a Large-scale Natural Language Text Processing System: CRITIQUE. In *Proceedings of the Second Conference on Applied Natural Language Processing*, ANLC '88, pages 195–202, Stroudsburg, PA, USA, 1988. Association for Computational Linguistics. doi: 10.3115/974235.974271. URL `https://doi.org/10.3115/974235.974271`.

Frankie Robertson. A Contrastive Evaluation of Word Sense Disambiguation Systems for Finnish. In *Proceedings of the Fifth International Workshop on Computational Linguistics for Uralic Languages*, pages 42–54, 2019.

Kay Rottmann and Stephan Vogel. Word reordering in statistical machine translation with a POS-based distortion model. In *Proceedings of TMI*, pages 171–180. Citeseer, 2007.

Lawrence K. Saul and Fernando Pereira. Aggregate and mixed-order Markov models for statistical language processing. *CoRR*, cmp-lg/9706007, 1997. URL `http://arxiv.org/abs/cmp-lg/9706007`.

Helmut Schmid. Part-of-speech Tagging with Neural Networks. In *Proceedings of the 15th Conference on Computational Linguistics - Volume 1*, COLING '94, pages 172–176, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. doi: 10.3115/991886.991915. URL `https://doi.org/10.3115/991886.991915`.

Bo Shu and Subhash Kak. A neural network-based intelligent metasearch engine. *Information Sciences*, 120:1–11, 1999.

William D Colen M Silva and Marcelo Finger. Improving CoGrOO: the Brazilian Portuguese Grammar Checker. In *Proceedings of the 9th Brazilian Symposium in Information and Human Language Technology*, 2013.

S. P. Singh, A. Kumar, L. Singh, M. Bhargava, K. Goyal, and B. Sharma. Frequency based spell checking and rule based grammar checking. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 4435–4439, March 2016. doi: 10.1109/ICEEOT.2016.7755557.

Jonas Sjöbergh. Chunking: an unsupervised method to find errors in text. In *Proceedings of the 15th Nordic Conference of Computational Linguistics, NODALIDA 2005*, 2005.

Jonas Sjöbergh. *The Internet as a Normative Corpus: Grammar Checking with a Search Engine.* Techical Report, Dept. of Theoretical Computer Science, Kungliga Tekniska Högskolan, 2009.

Madhvi Soni and Jitendra Singh Thakur. A Systematic Review of Automated Grammar Checking in English Language. *CoRR*, abs/1804.00540, 2018. URL `http://arxiv.org/abs/1804.00540`.

Sara Stymne and Lars Ahrenberg. Using a Grammar Checker for Evaluation and Postprocessing of Statistical Machine Translation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA). ISBN 2-9517408-6-7.

Anna Sågvall Hein. Language Control and Machine Translation. In *Proceedings of 7th International Conference on Theoretical and Methodological Issues in Machine Translation*, 10 1997.

Cornelia Tschichold. Evaluating second language grammar checkers. *Revue Tranel (Travaux neuchâtelois de linguistique)*, 21:195–204, 1994.

Cornelia Tschichold, Franck Bodmer, Etienne Cornu, Francois Grosjean, Lysiane Grosjean, Natalie Kubler, Nicolas Lewy, and Corinne Tschumi. Developing a new grammar checker for English as a second language. In

From Research to Commercial Applications: Making NLP Work in Practice, 1997.

Atro Voutilainen, Tanja Purtonen, and Kristiina Muhonen. *FinnTreeBank2 Manual*. University of Helsinki, Department of Modern Languages, 2012.

Ralph Weischedel, Richard Schwartz, Jeff Palmucci, Marie Meteer, and Lance Ramshaw. Coping with Ambiguity and Unknown Words Through Probabilistic Models. *Computational Linguistics*, 19(2):361–382, June 1993. ISSN 0891-2017. URL `http://dl.acm.org/citation.cfm?id=972470.972477`.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. Neural Language Correction with Character-Based Attention. *CoRR*, abs/1603.09727, 2016. URL `http://arxiv.org/abs/1603.09727`.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 619–628, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1057. URL `https://www.aclweb.org/anthology/N18-1057`.